

MultiReg-FE: Registered FE for Unbounded Inner-Product and Attribute-Weighted Sums

Qiuyan Du¹, Qianhan Chu¹, Jie Chen¹, Man Ho Au², and Debiao He³

¹ Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai 200062, China
s080001@e.ntu.edu.sg

² Department of Computing, The Hong Kong Polytechnic University
mhaau@polyu.edu.hk

³ School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Abstract. Recently, Francati et al. (Asiacrypt 2023) provided the first registered functional encryption (Reg-FE) beyond predicates. Reg-FE addresses the key escrow problem in functional encryption by allowing users to generate their own key pairs, effectively replacing the traditional private-key generator with a key curator. The key curator holds no secret information and runs deterministic algorithms to generate master public key for encryption and helper keys for decryption. However, existing Reg-FE schemes under standard assumptions require fixed data sizes, which limits their practicality in real-world applications.

In this work, we introduce Multi-Function Registered Functional Encryption for Inner-Product (MultiReg-FE for IP), a novel extension of Reg-FE. It enables users to register multiple functions under a single public key. With MultiReg-FE, we achieve both Reg-FE for Unbounded Inner-Product (Unbounded IP), which removes the need to predetermine vector lengths, and Reg-FE for Attribute-Weighted Sums with Inner-Product (AWSw/IP), allowing computations over arbitrary numbers of attribute-value pairs.

All our schemes achieve adaptive-IND-security. Specifically, we present:

- MultiReg-FE for Inner-Product, which supports unbounded number of function vectors from each user.
- Reg-FE for Unbounded Inner-Product, removing the need for preset vector lengths.
- The first Reg-FE for AWSw/IP in public-key settings.

Keywords: Registered Functional Encryption · Attribute-Weighted Sums · Unbounded Inner-Product.

1 Introduction

Functional Encryption (FE) [4,15] has emerged as a powerful cryptographic tool that allows fine-grained access control over encrypted data beyond “all-or-nothing” type. Instead of revealing entire plaintexts, FE enables users to compute specific functions on encrypted data, thus ensuring privacy while still allowing

useful computations. However, traditional FE schemes often suffer from a major limitation: the key escrow problem.

Registered Functional Encryption (Reg-FE). Reg-FE [10,7,20,6] was introduced to tackle the key escrow problem by allowing users to generate their own key pairs. Reg-FE eliminates the need for a trusted key authority while still providing fine-grained access control. Specifically, the system only requires a one-time trusted setup to generate a common reference string (CRS) crs, after which users can independently generate their public and secret keys (pk, sk) and register their public keys pk associated with the functions f with a key curator. The key curator only runs deterministic algorithms and holds no secret information. Upon the registration, the key curator updates the master public key mpk and distributes helper keys hsk to users, enabling them to decrypt specific functions of encrypted data.

While Reg-FE addresses key escrow, existing schemes are limited to requiring a predetermined size or structure of data. However, many real-world applications require more flexibility, particularly when dealing with uncertain or dynamically sized data. For example, in decentralized environments where the size of the data is not known in advance, predefined limits on vector lengths or function complexity can be impractical.

Our work is motivated by the need to handle dynamic and unbounded data. Specifically, we aim to extend Reg-FE to support unbounded inner-product (IP) functionalities and more complex operations like Attribute-Weighted Sums (with Inner-Product) (AWSw/IP), both of which can deal with dynamically sized datasets.

Unbounded Inner-Product Functionality. In a traditional Functional Encryption system for Inner-Product, the vector length must be determined during the setup phase. Therefore, a large length is usually set to ensure that various data can be handled, which can lead to inefficiencies when the data scale is unknown or dynamic. To solve this problem, Tomida and Takashima [17] proposed the first unbounded IPFE schemes from standard assumptions, where there is no need to fix the lengths of vectors in the setup phase and it can handle unbounded polynomial lengths of vectors. To handle vectors of different lengths, they used a method that, regardless of the length of a plaintext (or function) vector \mathbf{x} (or \mathbf{y}), splits it into multiple one-dimensional vectors $\{(x_i)\}_i$ (or $\{y_i\}_i$). This reduces the problem to handling fixed-length vectors. To ensure that only the sum $\sum_i x_i y_i$, they appended random values (whose sum equals zero) to the function vectors $\{(y_i)\}_i$ and a constant 1 to the plaintext vectors $\{(x_i)\}_i$. This ensures only the sum $\sum_{i \in [n]} (x_i, \rho_i)(y_i, 1)^\top = \sum_i x_i y_i = \mathbf{x}\mathbf{y}^\top$ is recovered, hiding individual products. This is a common approach to achieve unbounded inner-product. Except for inner-product, FE schemes for other unbounded functionalities (e.g., quadratic functions and predicate inner-product, etc.) have also been studied [8,16].

In addition to supporting arbitrary vector lengths, Reg-FE scheme for unbounded IP enables the reuse of the common reference string crs, which significantly reduces the complexity of the system. In traditional Reg-FE schemes, the

crs is tightly coupled to the data size, making crs reuse difficult and requiring a new trusted setup for each instance. However, Reg-FE for unbounded IP allows the same crs to be reused across multiple instances by decoupling the crs from the specific data parameters. In decentralized environments, where performing multiple trusted setups can be impractical, crs reuse offers a practical advantage.

Attribute-Weighted Sums Functionality. Consider a database where data users are allowed to compute aggregate statistics on encrypted data, like average salaries or medical conditions, without compromising individual privacy. More precisely, the data are multiple attribute-value pairs $(\mathbf{x}_j, \mathbf{z}_j)$, f is a function given by a user. The owner of the database only allow the user to compute weighted sums $\sum_j f(\mathbf{x}_j)\mathbf{z}_j^\top$ without access to any other information of $\{\mathbf{z}_j\}_j$. This is where Attribute-Weighted Sums (AWS), a class of functionality proposed by [2], comes into play. AWS enables the computation of sums $\sum_j f(\mathbf{x}_j)\mathbf{z}_j^\top$ where each term is the product of a private value \mathbf{z}_i and a function f applied to a public value \mathbf{x}_i . It ensures that only the weighted sum is revealed during decryption, not the individual \mathbf{z}_i values.

Attribute-Weighted Sums (with Inner-Product) (AWSw/IP) proposed by [3] is an direct extension of AWS. In short, this new functionality supports the sum of regular AWS and additional inner product, that is $\sum_j f(\mathbf{x}_j)\mathbf{z}_j^\top + \mathbf{p}\mathbf{q}^\top$, where \mathbf{p} is a private vector from encryptor and \mathbf{q} is an additional function vector. Agrawal et al. [3] propose a generic method to transform FE for inner product (IPFE) to FE for AWSw/IP using the Partial Garbling Scheme (PGS) [13,18,2]. Specifically, PGS can transform one AWSw/IP function into multiple vector functions. But this method only supports the transformation from IPFE in secret-key setting.

This Work. We focus on the challenges of handling unbounded data and complex computations in a decentralized and non-interactive manner. We note that both unbounded IP and AWSw/IP require users to handle multiple function vectors simultaneously, making it necessary to design a flexible scheme that can accommodate this need. Thus, we introduce a auxiliary scheme with a novel registration pattern as our core technique, which allows a single public key to be associated with multiple functions. We refer to it as Multi-Function Reg-FE (MultiReg-FE). With this scheme, we can simply achieve Reg-FE for *unbounded IP*, which can handle arbitrary vector lengths, or *AWSw/IP* to support arbitrary number of attribute-value pairs, suitable for environments where data size and structure are not known in advance.

To illustrate how this auxiliary scheme works in practice, we now introduce the details of MultiReg-FE.

Multi-Function Registered Functional Encryption (MultiReg-FE). In MultiReg-FE, each user registers multiple functions, each labeled with a number corresponding to a specific function set. For instance, in a collaborative data analysis platform, researchers or organizations may need to compute different functions on encrypted datasets. These functions are grouped into categories,

and within each category (or function set), each researcher registers a unique function. For example, all researchers may register a function for category 1 (e.g., computing the mean), category 2 (e.g., performing a regression), and so on, up to M categories.

When a dataset is encrypted, the encryptor can choose a specific function set (or sets) for encryption. Users can then decrypt and compute the value of their registered function within that set. For example, if the dataset is encrypted for function set r , researcher i can only decrypt the data using their registered function $f_{i,r}$, obtaining $f_{i,r}(x)$, but not any functions from other sets or from other researchers.

This flexible registration pattern allows each user to associate a single public key with multiple functions, enabling efficient encryption and decryption without relying on a trusted central authority. Moreover, in our MultiReg-FE scheme, the number of function sets M need not be determined in the setup phase.

For inner-product, MultiReg-FE can be extended to support multiple input vectors. The encryptor can encrypt data vectors $\mathbf{x}_1, \dots, \mathbf{x}_N$ for several sets of function vectors $\{\mathbf{y}_{i,\phi(1)}\}_i, \dots, \{\mathbf{y}_{i,\phi(N)}\}_i$, where $\phi(\cdot)$ is a labeling function. User i can then compute only the sum of their registered function values, $\sum_{j \in [N]} \mathbf{x}_j \mathbf{y}_{i,\phi(j)}$.

1.1 Our Results

We present the following key contributions:

- **MultiReg-FE for Inner-Product:** We introduce Multi-Function Registered Functional Encryption (MultiReg-FE), where each user i registers a public key associated with multiple function vectors $\{\mathbf{y}_{i,r}\}_{r \in [m_i]}$, where m_i is arbitrary and does not need to be fixed during setup. An encryptor can encrypt multiple vectors $\{\mathbf{x}_j\}_{j \in [N]}$ for a set of function vectors $\{\mathbf{y}_{i,r}\}_{r \in \phi(N)}$ labeled by $\phi(\cdot)$; , and user i can only decrypt and retrieve the sum $\sum_{j \in [N]} \mathbf{x}_j \mathbf{y}_{i,\phi(j)}^\top$, provided $\phi([N]) \subseteq [m_i]$. This scheme achieves adaptive-IND-security.
- **Reg-FE for Unbounded Inner-Product:** With the MultiReg-FE scheme, we achieve the first registered functional encryption scheme supporting unbounded inner-product functionalities. This scheme allows for unbounded vector lengths, removing the need to predetermine vector sizes during the setup phase. This scheme also achieves adaptive-IND-security.
- **Reg-FE for Attribute-Weighted Sums (with Inner-Product):** We obtain the first Reg-FE for attribute-weighted sums (with inner-product) (AWSw/IP). This scheme allows for computations over an arbitrary number of attribute-value pairs. It is the first scheme to implement AWSw/IP functionality in public-key settings, whereas previous schemes for AWS were limited to private-key settings. This scheme also achieves adaptive-IND-security.

Related Works. Reg-FE for functionality beyond predicates is a new cryptographic primitive that has been studied recently. Both Francati et al. [10] and

Datta et al. [7] presented a Reg-FE scheme for general functionalities from indistinguishability obfuscation (iO) and binding hash functions, and their schemes support an exponential number of users. Zhu et al. [20] introduced the first Reg-FEs for linear functions (or inner-product) and for quadratic functions using pairings. The scheme for linear functions achieves adaptive-IND-security. Meanwhile, the scheme for quadratic functions achieves very-selective-SIM-security, a weaker security notion defined by Zhu et al., which requires adversaries to declare their challenge functions, and the types of challenge public keys before the setup phase. Simultaneously, Chu et al. [6] proposed a Reg-FE scheme for linear functions that achieves weakly selective-IND-security and weakly selective-SIM-security. These weakly selective security definitions, introduced by Chu et al. Specifically, weakly selective-IND security extends the selective-IND-security by requiring that for any function f in the function space, the challenge messages \mathbf{x}_0^* and \mathbf{x}_1^* satisfy $f(\mathbf{x}_0^*) = f(\mathbf{x}_1^*)$. Weakly selective-SIM security allows the adversary to obtain more function values in the final simulation game without revealing the encrypted message, which is considered acceptable in centralized FE where the message remains hidden. Zhang et al. proposed bounded collision-resistant Reg-FE schemes for general functionalities without iO, achieving adaptive-SIM-security. Compared to plain RFE, bounded collision-resistant Reg-FE additionally requires that a prior-bound Q of the number of corrupted users should be declared at the setup phase. We summarize our results and existing Reg-FE with full collusion-resistance and from standard assumptions in Table 1.

Scheme	Function	Assumption	Security	Unbounded
ZL24-1 [20]	Linear	k -Lin	<i>adp</i> -IND	×
ZL24-2 [20]	Quadratic	bi- k -Lin	<i>sel</i> *-SIM	×
CL24-1[6]	Linear	k -Lin	<i>sel</i> '-IND	×
CL24-2[6]	Quadratic	bi- k -Lin	<i>sel</i> '-SIM	×
Our (3)	Linear	k -Lin	<i>adp</i> -IND	✓
Our (4)	Linear	k -Lin	<i>adp</i> -IND	✓
Our (5)	AWSw/IP	k -Lin	<i>adp</i> -IND	✓

Table 1. Comparison with all existing schemes achieving full collusion-resistance from standard assumptions. In the column of **Function**, “Linear” stands for “linear functions (inner-product)”. In the column of **Security**, *adp*, *sel*, *sel**, and *sel*' stand for “adaptive”, “selective”, “very selective” [20], and “weakly selective” [6], respectively; IND and SIM stand for “indistinguishability-based security” and “simulation-based security”, respectively. In the column of **Unbounded**, schemes (3), (4) and (5) support unbounded “number of functions with one public key”, “vector length” and “number of attribute-value pairs”, respectively.

1.2 Technical Overview

In this part, we focus on constructing the slotted Reg-FE. There is already a proven and mature method to convert slotted Reg-FE to Reg-FE using the ‘powers-of-two’ transformation [11,12,10,20].

Here, we briefly introduce the concept of slotted Reg-FE, a simplified version of Reg-FE, where the number of registered users is fixed and all users must register their public keys with the curator together. This eliminates the need to handle updates for newly registered users. Specifically, a L -slotted Reg-FE scheme, which has L registered users, consists of six algorithms (Setup, Gen, Ver, Agg, Enc, Dec). The system relies on one-time trusted sampling to generate a common reference crs only at Setup, and the key generation algorithm is run by every user to obtain their own key pair (pk, sk) . There are only L users to register their public keys pk and functions f with the curator. After receiving all L pairs of public key and function (pk, f) , the curator needs to verify that each public key is correctly generated using the verification algorithm Ver. Then the curator aggregates the pairs $\{(pk_i, f_i)\}_{i \in [L]}$ to produce a master public key mpk and L helper keys hsk_1, \dots, hsk_L for the corresponding users. The master public key mpk is for encryptors to encrypt their data x . Each registered user, who has registered a different function f , can decrypt the data to obtain the value of their specific function $f(x)$ using their secret key sk and helper key hsk .

To extend this to a more flexible setting, we introduce MultiReg-FE, where a single public key can be associated with multiple functions. Specifically, in MultiReg-FE for IP, each user generates a single public-secret key pair and registers multiple vectors with the key curator, all under the same public key.

Starting Point: Slotted MultiReg-FE. To associate a public key with multiple functions (e.g., M functions), the first idea is to simply extend the length of the vector to M times the original length. However, when the system is set up, M is fixed, making it impossible to register a public key associated with an arbitrary number of functions.

Thus, we need to find another way to associate a public key with multiple function vectors.

Our construction idea is quite simple. It can be seen as a multi-instance extension of Zhu et al.’s work, but with a shared common reference string crs across all instances.

Here, we briefly recap the core components of the L -slotted Reg-FE scheme for inner products of vector length n . For $i \in [L]$,

$$\begin{aligned}
\text{crs} &= ([\mathbf{A}]_1, \{[\mathbf{B}_1 \mathbf{r}_i^\top]_2, [\mathbf{R}_i, \mathbf{A}\mathbf{W}_i]_1\}_{i \in [L]}, \{[\mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2\}_{i \in [L], u \in [L] \setminus \{i\}}) \\
\text{pk}_i &= ([\mathbf{A}\mathbf{U}_i, \mathbf{R}_i \mathbf{U}_i]_1, \{[\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus \{i\}}) \\
\text{sk}_i &= \mathbf{U}_i \\
\text{mpk} &= [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]_1 \\
&= [\mathbf{A}, \sum_u (\mathbf{A}\mathbf{U}_u + \mathbf{A}\mathbf{W}_u (\mathbf{y}_u^\top \otimes \mathbf{I}_{2k+1})), \sum_u \mathbf{A}\mathbf{W}_u]_1 \\
\text{hsk}_i &= [\mathbf{k}_{i,0}^\top, \mathbf{k}_{i,1}^\top, \mathbf{K}_{i,2}]_2 \\
&= [\mathbf{B}_1 \mathbf{r}_i^\top, \sum_{u \neq i} (\mathbf{U}_u \mathbf{B}_1 \mathbf{r}_i^\top + \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}_u^\top), \sum_{u \neq i} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2 \\
\text{ct} &= [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3]_1 \\
&= [\mathbf{s}\mathbf{P}_0, \mathbf{s}\mathbf{P}_1, \mathbf{x} \otimes \mathbf{v} + \mathbf{s}\mathbf{P}_2, \mathbf{v}]_1 \\
&= [\mathbf{s}\mathbf{A}, \mathbf{s} \sum_{u \in [L]} (\mathbf{A}\mathbf{U}_u + \mathbf{A}\mathbf{W}_u (\mathbf{y}_u^\top \otimes \mathbf{I}_{2k+1})), \mathbf{x} \otimes \mathbf{v} + \mathbf{s} \sum_{u \in [L]} \mathbf{A}\mathbf{W}_u, \mathbf{v}]_1
\end{aligned}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}$, $\mathbf{B} \leftarrow \mathbb{Z}_p^{(2k+1) \times k}$, $\mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+1)}$, $\mathbf{W}_i \leftarrow \mathbb{Z}_p^{(2k+1) \times n(2k+1)}$, $\mathbf{U}_i \leftarrow \mathbb{Z}_p^{(2p+1) \times (2p+1)}$, $\mathbf{s} \leftarrow \mathbb{Z}_p^{1 \times k}$ and $\mathbf{v} \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$.

To construct MultiReg-FE, consider a simple case where we have M sets of function vectors $\{\mathbf{y}_{i,1}\}_{i \in [L]}, \dots, \{\mathbf{y}_{i,M}\}_{i \in [L]}$, and N plaintext vectors $\mathbf{x}_1, \dots, \mathbf{x}_N$. We extend the original scheme by replicating the components related to the function vectors and plaintext vectors. Specifically, we extend the terms corresponding to the function vectors (e.g., \mathbf{P}_1 and $\mathbf{k}_{i,1}^\top$) to handle multiple sets of function vectors $\{\mathbf{y}_{i,1}\}_{i \in [L]}, \dots, \{\mathbf{y}_{i,M}\}_{i \in [L]}$, and likewise extend the terms corresponding to the plaintext vectors (e.g., $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$) to process multiple plaintexts $\mathbf{x}_1, \dots, \mathbf{x}_N$.

Next, we need to ensure that decryption reveals only the sum $\sum_{j \in [N]} \mathbf{x}_j \mathbf{y}_{i, \phi(j)}^\top$ without leaking any other information. As we mentioned before, Tomida et al. [17] achieved this by appending random values (whose sum equals zero) to the function vectors, which relies on Private Key Generator (acting as a trusted center) to sample these random values. In contrast, there is no trusted center in our scheme. Instead, we delegate the role of sampling random values to the encryptor. Specifically, we append an additional random component ρ_j to each vector \mathbf{x}_j and require $\sum_{j \in [N]} \rho_j = 0$. Correspondingly, we need to add an additional component 1 to each vector \mathbf{y}_i to match the dimension of the vector \mathbf{x}_i .

Then, we can consider the case where users registers different number of functions. Let m_i be the number of functions registered by user i . We can just let $M = \max_{i \in [L]} m_i$. Then for the case where the number of function vectors m_i is less than M , we make up to M with zero vectors, i.e. set $\mathbf{y}_{i,r} = \mathbf{0}$ for $i \in [L]$, $r > m_i$. In this case, user i can't decrypt the ciphertext ct associated with the labeling function $\phi(\cdot)$ if $\phi([N]) \not\subseteq [m_i]$.

Finally, our overall extension is structured as follows, with boxing the changes.

$$\begin{aligned} \mathbf{P}_{1,r} &\rightarrow \{\mathbf{P}_{1,r}\}_{r \in [M]} \\ \mathbf{k}_{i,1}^\top &\rightarrow \{\mathbf{k}_{i,1,r}^\top\}_{r \in [M]} \\ \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2 &\rightarrow \{\mathbf{c}_{0,j}, \mathbf{c}_{1,j}, \mathbf{c}_{2,j}\}_{j \in [N]}, \phi(\cdot) \end{aligned}$$

where

$$\begin{aligned} \mathbf{P}_{1,r} &= \sum_{u \in [L]} (\mathbf{A}\mathbf{U}_u + \mathbf{A}\mathbf{W}_u (\boxed{(\mathbf{y}_{u,r}, 1)^\top} \otimes \mathbf{I}_{2k+1})) \\ \mathbf{k}_{i,1,r}^\top &= \sum_{u \in [L] \setminus \{i\}} (\mathbf{U}_u \mathbf{B}_1 \mathbf{r}_i^\top + \mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \boxed{(\mathbf{y}_{u,r}, 1)^\top}) \\ \mathbf{c}_{0,j} &= \mathbf{s}_j \mathbf{P}_0 \\ &= \mathbf{s}_j \mathbf{A} \\ \mathbf{c}_{1,j} &= \mathbf{s}_j \mathbf{P}_{1,\phi(j)} \\ &= \sum_{u \in [L]} (\mathbf{s}_j \mathbf{A}\mathbf{U}_u + \mathbf{s}_j \mathbf{A}\mathbf{W}_u (\mathbf{y}_{u,\phi(j)}^\top \otimes \mathbf{I}_{2k+1})) \\ \mathbf{c}_{2,j} &= \boxed{(\mathbf{x}_j, \rho_j)} \otimes \mathbf{v} + \mathbf{s}_j \mathbf{P}_2 \\ &= \boxed{(\mathbf{x}_j, \rho_j)} \otimes \mathbf{v} + \mathbf{s}_j \sum_{u \in [L]} \mathbf{A}\mathbf{W}_u \end{aligned}$$

and $\phi(\cdot)$ is a labeling function, $\mathbf{v} \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, $\mathbf{s}_1, \dots, \mathbf{s}_N \leftarrow \mathbb{Z}_p^{1 \times k}$, $\rho_1, \dots, \rho_N \leftarrow \mathbb{Z}_p$ s.t. $\sum_{j \in [N]} \rho_j = 0$.

Although our construction method is conceptually simple, the security proof is far from trivial. We give the full proof in Section 3.2. The basic proof strategy is similar to that of Zhu et al. [20], but we face several new challenges that do not arise in the single-instance case.

Challenges in the Proof. Here, we highlight two of the primary technical challenges encountered in our security proof:

1. Dealing with Malicious Public Keys in a Multi-Instance Setting.

The first challenge arises when we need to process multiple plaintext vectors simultaneously. Specifically, malicious users may register a malicious public key pk that is not generated by the trusted key generation algorithm Gen . In such cases, the challenger lacks knowledge of the corresponding secret key \mathbf{U}_i , rendering the malicious public key indistinguishable from random values rather than the legitimate form $([\mathbf{A}\mathbf{U}_i, \mathbf{R}_i \mathbf{U}_i]_1, \{[\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus \{i\}})$. Thus, we use the Non-Interactive Zero-Knowledge (NIZK) Arguments in the algorithm Ver to verify the correctness of a malicious public key pk . Once the key is verified, it holds that $\mathbf{T}_i = \mathbf{A}\mathbf{U}_i$ and $\mathbf{Q}_i = \mathbf{R}_i \mathbf{U}_i$ for some unknown \mathbf{U}_i .

The challenge, however, is that in the ciphertext, every $[\mathbf{s}_j \mathbf{A}]_1$ must be treated as random value $[\mathbf{a}_j]_1$ for all j , which can be proved by MDDH assumption. Note that the value $\mathbf{s}_j \mathbf{T}_i$ contains $\mathbf{s}_j \mathbf{A}$, so we need to find a way to eliminate the influence of $\mathbf{s}_j \mathbf{A}$ embedded in $\mathbf{s}_j \mathbf{T}_i$ so that we can use MDDH assumption.

Previous works, such as [21,20], implicitly set the structure of \mathbf{R}_i for all i as:

$$\widehat{\mathbf{R}}_i = \widetilde{\mathbf{R}}_i \begin{pmatrix} \mathbf{s}\mathbf{A} \\ \mathbf{I}_{2k+1} \end{pmatrix}, \text{ where } \widetilde{\mathbf{R}}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+2)}$$

They replaced \mathbf{sT}_i with $\mathbf{e}_1 \tilde{\mathbf{R}}_i^{-1} \mathbf{Q}_i$ to remove the influence of \mathbf{sT}_i . However, this method is effective only in a single-instance setting, where there is only one \mathbf{s} . Because we can neither replace different $\{\mathbf{s}_j \mathbf{A}\}_j$ with the same $\mathbf{e}_1 \tilde{\mathbf{R}}_i^{-1} \mathbf{Q}_i$ nor embed all distinct $\{\mathbf{s}_j \mathbf{A}\}_j$ in \mathbf{R}_i .

Our approach is different: instead of treating $\mathbf{s}_j \mathbf{T}_i$ as a whole, we explicitly extract \mathbf{A} from \mathbf{T}_i . To achieve this, we propose a conceptual change to the structure of $\hat{\mathbf{R}}_i$:

$$\hat{\mathbf{R}}_i = \begin{pmatrix} \tilde{\mathbf{R}}_i \\ \mathbf{t}_i \end{pmatrix}$$

where $\tilde{\mathbf{R}}_i \leftarrow \mathbb{Z}_p^{(2k+1) \times (2k+1)}$ and $\mathbf{t}_i \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$. This allows us to express \mathbf{T}_i as:

$$\mathbf{T}_i = \mathbf{A} \tilde{\mathbf{R}}_i^{-1} (\mathbf{I}_{2k+1} \parallel \mathbf{0}^\top) \mathbf{Q}_i,$$

where $\mathbf{0}^\top$ is a zero vector in \mathbb{Z}_p^{2k+1} . Thus, we can express $\mathbf{s}_j \mathbf{A}$ as $\mathbf{s}_j \mathbf{A} \tilde{\mathbf{R}}_i^{-1} (\mathbf{I}_{2k+1} \parallel \mathbf{0}^\top) \mathbf{Q}_i$ for all j , allowing us to handle multiple plaintext vectors.

2. Generating Random Values to Hide an Unbounded Number of Plaintext Vectors with a Shared CRS. The second challenge involves generating random values to hide an unbounded number of plaintext vectors when using a shared CRS. This is crucial in the nested dual system methodology, where the hidden plaintext vectors are finally replaced with random vectors.

In prior works such as [21,20], the approach was limited to handling a single plaintext vector (i.e. $N = 1$), because they use the following argument to sample randomness

$$\left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_i]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_i^\top, \mathbf{A} \mathbf{W}_i, \mathbf{W}_i (\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_i (\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \{[\mathbf{a}_j]_1, [\mathbf{a}_j \mathbf{U}_i + \mathbf{a}_j \mathbf{W}_i]_1, [\mathbf{a}_j \mathbf{W}_i]_1\}_{j \in [N]} \\ \mathbf{A} \mathbf{U}_i, [\mathbf{R}_i \mathbf{U}_i]_1, \mathbf{U}_i \mathbf{B}_1, \mathbf{U}_i \mathbf{B}_3 \end{array} \right\} \approx_c \left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_i]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_i^\top, \mathbf{A} \mathbf{W}_i, \mathbf{W}_i (\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_i (\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \{[\mathbf{a}_j]_1, [\mathbf{a}_j \mathbf{U}_i + c_{j,i} \mathbf{s}^{(3)} + \mathbf{a}_j \mathbf{W}_i + \mathbf{w}_{j,i} \otimes \mathbf{s}^{(3)}]_1, [\mathbf{a}_j \mathbf{W}_i + \mathbf{w}_{j,i} \otimes \mathbf{s}^{(3)}]_1\}_{j \in [N]} \\ \mathbf{A} \mathbf{U}_i, [\mathbf{R}_i \mathbf{U}_i + \mathbf{u}_i \mathbf{s}^{(3)}]_1, \mathbf{U}_i \mathbf{B}_1, \mathbf{U}_i \mathbf{B}_3 \end{array} \right\}$$

However, in this argument, N must be 1. Simply put, they applied transformations such as $\mathbf{W}_i \rightarrow \mathbf{W}_i + \mathbf{a}_1^\perp (\mathbf{w}_{1,i} \otimes \mathbf{s}^{(3)})$ and $\mathbf{U}_i \rightarrow \mathbf{U}_i + \mathbf{a}_1^\perp (\mathbf{u}_i \mathbf{s}^{(3)})$, where $\mathbf{a}_1 \mathbf{a}_1^\perp = 1$, a technique that relies on dual vectors. However, in the multi-instance setting (i.e., when $N \neq 1$), this technique becomes difficult to apply, because a single \mathbf{a}_1^\perp cannot satisfy $\mathbf{a}_j \mathbf{a}_1^\perp = 1$ for all $j \in [N]$. Therefore, this method fails to handle multiple plaintext vectors.

To overcome this limitation, we are inspired by the Entropy Expansion Lemma presented in [5]. We introduce novel lemmas 3 and 4 that rely on two DDH-like (Decisional Diffie-Hellman) assumptions on matrices. These lemmas allow us to generate an unbounded number of random values, which are necessary to hide multiple plaintext vectors effectively.

To Slotted Reg-FE for Unbounded IP. We extend Reg-FE for unbounded inner products by leveraging the registration pattern of MultiReg-FE. First, the

Setup algorithm in slotted MultiReg-FE is independent of the number of functions registered by each user. Second, an arbitrary number of function vectors can naturally extend to vectors of unbounded length. To achieve this, for any vector $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,m_i}) \in \mathbb{Z}_p^{m_i}$, we treat each element $y_{i,r}$ as a single-entry vector $\mathbf{y}_{i,r} = (y_{i,r}) \in \mathbb{Z}_p^1$ for all $r \in [m_i]$. Similarly, for $\mathbf{x} \in \mathbb{Z}_p^N$, we generate $\{\mathbf{x}_j\}_{j \in [N]}$, where each \mathbf{x}_j is a 1-dimensional vector. Finally, we use $\{\mathbf{y}_{i,r}\}_{r \in [m_i]}$ for all i and $\{\mathbf{x}_j\}_{j \in [N]}$ as the function vectors and plaintext vectors in the MultiReg-FE scheme.

To Slotted Reg-FE for AWSw/IP. Along the way, we also propose a slotted Reg-FE for AWSw/IP. AWSw/IP proposed by [2,3] is a direct extension of AWS. In FE for AWSw/IP, an arithmetic branching program f and a vector \mathbf{q} are associated with a public key; multiple attribute-value pairs $\{(\mathbf{x}_j, \mathbf{z}_j)\}_j$ and a vector \mathbf{p} are associated with ciphertext, where only $\{\mathbf{x}_j\}_i$ are public and $\{\mathbf{z}_j\}_i$; and decryption only reveals $\mathbf{p}\mathbf{q}^\top + \sum_i f(\mathbf{x}_j)\mathbf{z}_j^\top$.

Agrawal et al. [3] proposed a generic method to transforming FE for IP to FE for AWSw/IP via a partial garbling scheme (PGS) [13,18,2]. This method inherently requires multiple functions to be specified per user, which aligns with the registration pattern of MultiReg-FE. However, their approach is confined to the private-key setting, where there is a trusted center to generate secret encryption key and distribute it to encryptors. The primary reason for this limitation is that their method relies on the generation of random vectors $\{\mathbf{t}_j\}_j$ by a trusted center to hide private information $\{\mathbf{z}_j\}_j$. Consequently, their method does not extend to settings where decentralized key management is required, such as in registration-based systems without a trusted center.

The main challenge lies in eliminating the necessity for the center to sample these random values. We scrutinize the algebraic structure of their generic transformations and propose the following method suitable for no-trusted-center systems. And we notice a similar idea is used in [19,16].

For function $\{f_i \in \mathcal{F}_{\text{ABP},n_1,n_2}\}_{i \in [L]}$, we compute $\mathbf{L}_{i,0}$ and $\mathbf{L}_{i,1} = \{\mathbf{L}_{i,1,1}, \dots, \mathbf{L}_{i,1,n}\}$ from f_i , and set

$$\mathbf{L}_i = \begin{Bmatrix} \mathbf{L}_{i,1,1} \\ \dots \\ \mathbf{L}_{i,1,n} \\ \mathbf{L}_{i,0} \end{Bmatrix} = \{\ell_{i,1}, \dots, \ell_{i,m}\}_{(n_1+1)(m+n_2-1) \times m}.$$

We set the form of function vectors and plaintext vector as follows.

$$\begin{aligned} \text{(function vectors)} \quad \tilde{\mathbf{y}}_{i,r} &= \begin{cases} (\ell_{i,r}^\top, 0^{n_2}, 1, \mathbf{q}_i) & \text{if } r = 1 \\ (\ell_{i,r}^\top, 0^{n_2}, 0, 0^{n_3}) & \text{if } 1 < r \leq m \\ (0^{m'}, \mathbf{e}_{k-s}, 0, 0^{n_3}) & \text{if } m < r \leq M \end{cases} \\ \text{(input vectors)} \quad \tilde{\mathbf{x}}_j &= \begin{cases} ((\mathbf{x}_j, 1) \otimes \boxed{\mathbf{t}_j}, \mathbf{z}_j - \boxed{\bar{\mathbf{t}}_j}, w_j, \mathbf{p}) & \text{if } j = 1 \\ ((\mathbf{x}_j, 1) \otimes \boxed{\mathbf{t}_j}, \mathbf{z}_j - \boxed{\bar{\mathbf{t}}_j}, w_j, 0^{n_3}) & \text{if } j \in [N] \setminus \{1\} \end{cases} \end{aligned}$$

where $\sum_{j \in [N]} w_j = 0$. Note that, as indicated by the boxed terms, we have already shifted the role of sampling random vectors $\{\mathbf{t}_j\}_j$ to the encryptor. Clearly, for $j = 1$,

$$(\tilde{\mathbf{y}}_{i,1}\tilde{\mathbf{x}}_j, \dots, \tilde{\mathbf{y}}_{i,M}\tilde{\mathbf{x}}_j) = \text{pgb}^+(f, \mathbf{x}_j, \mathbf{z}_j, w_j + \mathbf{p}\mathbf{q}_i);$$

for $j \in [N] \setminus \{1\}$,

$$(\tilde{\mathbf{y}}_{i,1}\tilde{\mathbf{x}}_j, \dots, \tilde{\mathbf{y}}_{i,M}\tilde{\mathbf{x}}_j) = \text{pgb}^+(f, \mathbf{x}_j, \mathbf{z}_j, w_j).$$

2 Preliminaries

Notations. For a finite set R , we use $|R|$ to denote its size and denote by $r \leftarrow R$ the fact that r is picked uniformly at random from a finite set R . We denote $[L]$ the set $\{1, \dots, L\}$. For an ordered list or array \mathcal{D} , we denote $\mathcal{D}[i]$ the i -th element of \mathcal{D} . When $|\mathcal{D}| < i$ or $i < 1$, we denote $\mathcal{D}[i] = \perp$; when we append d to \mathcal{D} , we set $\mathcal{D}[|\mathcal{D} + 1|] = d$. We denote \star and \cdot as wildcards. We use $\text{negl}(\lambda)$ to denote a negligible function of λ . We use \approx_s to denote two distributions being statistically indistinguishable, and use \approx_c to denote two distributions being computationally indistinguishable. We use bolded lowercase to denote row vectors (e.g. \mathbf{x}) and use bolded uppercase to denote matrices (e.g. \mathbf{A}). We use $\text{span}(\mathbf{A})$ to denote the row span of \mathbf{A} , and use $\text{basis}(\mathbf{A})$ to denote a basis of the column space of \mathbf{A} . Let \mathbb{F} be a field. We denote $\mathbf{A} \otimes \mathbf{B}$ a Kronecker Product for matrices $\mathbf{A} \in \mathbb{F}^{k \times n}$ and $\mathbf{B} \in \mathbb{F}^{m \times \ell}$. For matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ of proper sizes, we have $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$. For simplicity, we use $\{c_i\}_i$ to denote $\{c_i\}_{i \in [N]}$ in the overly long equation.

2.1 Prime-Order Groups

A generator \mathcal{G} takes as input a security parameter λ and outputs a description $\mathbb{G} := (p, G_1, G_2, G_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, G_1, G_2 and G_T are cyclic groups of order p , and $e : G_1 \times G_2 \rightarrow G_T$ is a non-degenerate bilinear map. We require that the group operations in G_1, G_2 and G_T as well the bilinear map e are computable in deterministic polynomial time with respect to λ . Let $g_1 \in G_1, g_2 \in G_2$ and $g_T = e(g_1, g_2) \in G_T$ be the respective generators. We employ the implicit representation of group elements: for a matrix \mathbf{M} over \mathbb{Z}_p , we define $[\mathbf{M}]_1 := g_1^{\mathbf{M}}, [\mathbf{M}]_2 := g_2^{\mathbf{M}}, [\mathbf{M}]_T := g_T^{\mathbf{M}}$, where exponentiation is carried out component-wise. Also, given $[\mathbf{A}]_1, [\mathbf{B}]_2$, we let $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. We define the matrix decision Diffie-Hellman (MDDH) assumption on G_1 [9]:

Definition 1 (MDDH $_{k,\ell}^m$ Assumption [9]). *Let $\ell > k \geq 1$ and $m \geq 1$. We say that the MDDH $_{k,\ell}^m$ assumption holds if for all PPT adversaries \mathcal{A} , the following advantage function is negligible in λ .*

$$\text{Adv}_{\mathcal{A}}^{\text{MDDH}_{k,\ell}^m}(\lambda) := |\Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{MS}]_1) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{U}]_1) = 1]|$$

where $\mathbf{M} \leftarrow_{\mathbb{R}} \mathbb{Z}_p^{\ell \times k}, \mathbf{S} \leftarrow_{\mathbb{R}} \mathbb{Z}_p^{k \times m}$ and $\mathbf{U} \leftarrow_{\mathbb{R}} \mathbb{Z}_p^{\ell \times m}$.

The MDDH assumption on G_2 can be defined in an analogous way. Escala et al. [9] showed that

$$k\text{-Lin} \Rightarrow \text{MDDH}_{k,k+1}^1 \Rightarrow \text{MDDH}_{k,\ell}^m \quad \forall \ell > k, m \geq 1$$

with a tight security reduction.

2.2 Dual System

This part is mainly adopted from [20].

Let $\ell_1, \ell_2, \ell_3 \geq 1$ and $\ell := \ell_1 + \ell_2 + \ell_3$. We use basis

$$\mathbf{B}_1 \leftarrow \mathbb{Z}_p^{\ell \times \ell_1}, \mathbf{B}_2 \leftarrow \mathbb{Z}_p^{\ell \times \ell_2}, \mathbf{B}_3 \leftarrow \mathbb{Z}_p^{\ell \times \ell_3}$$

we denote $\mathbf{B}_1^\parallel, \mathbf{B}_2^\parallel, \mathbf{B}_3^\parallel$ as its dual basis, for all $\sigma, \delta \in \{1, 2, 3\}$, it holds that:

$$\mathbf{B}_\sigma^\top \mathbf{B}_\delta^\parallel = \begin{cases} \mathbf{I} & \text{when } \sigma = \delta \text{ (non-degeneracy)} \\ \mathbf{0} & \text{when } \sigma \neq \delta \text{ (orthogonality)} \end{cases}$$

Facts. With basis $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$ and its dual basis $\mathbf{B}_1^\parallel, \mathbf{B}_2^\parallel, \mathbf{B}_3^\parallel$, for all $\mathbf{t} \in \mathbb{Z}_p^{1 \times n\ell}$, we can uniquely decompose \mathbf{t} as

$$\mathbf{t} = \sum_{\sigma \in \{1,2,3\}} \mathbf{t}^{(\sigma)} \quad \text{where } \mathbf{t}^{(\sigma)} \in \text{span} \left(\mathbf{I}_n \otimes \left(\mathbf{B}_\sigma^\parallel \right)^\top \right)$$

Note that for all $\sigma \in \{1, 2, 3\}$ and $n \in \mathbb{N}$, $\mathbf{t}^{(\sigma)}$ can be seen as the projection of \mathbf{t} onto $\text{span} \left(\mathbf{I}_n \otimes \left(\mathbf{B}_\sigma^\parallel \right)^\top \right)$, and for each $S \subseteq \{1, 2, 3\}$, we write $\mathbf{t}^S = \sum_{\sigma \in S} \mathbf{t}^{(\sigma)}$. Moreover, it holds that:

$$\mathbf{v} \mathbf{B}_\sigma = \mathbf{t}^{(\sigma)} \mathbf{B}_\sigma, \quad \text{and} \quad \{\mathbf{t}^{(\sigma)}, \{\mathbf{t}^{(\delta)}\}_{\delta \neq \sigma}\} \equiv \{\mathbf{t}^*, \{\mathbf{t}^{(\delta)}\}_{\delta \neq \sigma}\}$$

where $\mathbf{t}^* \leftarrow \text{span} \left(\mathbf{I}_n \otimes \left(\mathbf{B}_\sigma^\parallel \right)^\top \right)$

Definition 2 (SD $_{\mathbf{B}_1 \rightarrow \mathbf{B}_2}^{\mathbb{G}_s}$ Assumption). $s \in \{1, 2\}$, Let $\ell_1, \ell_2, \ell_3 \geq 1$ and $\ell := \ell_1 + \ell_2 + \ell_3$. We say that the subspace decision assumption $\text{SD}_{\mathbf{B}_1 \rightarrow \mathbf{B}_2}^{\mathbb{G}_s}$ holds in \mathbb{G}_s if there exist an efficient sampler outputting random $[\mathbf{B}_1]_s \in \mathbb{G}_s^{\ell \times \ell_1}$, $[\mathbf{B}_2]_s \in \mathbb{G}_s^{\ell \times \ell_2}$, $[\mathbf{B}_3]_s \in \mathbb{G}_s^{\ell \times \ell_3}$ along with its dual basis: $\mathbf{B}_1^\parallel, \mathbf{B}_2^\parallel, \mathbf{B}_3^\parallel$ such that for all PPT adversaries \mathcal{A} , the following advantage function is negligible in λ .

$$\text{Adv}_{\mathcal{A}, s, \ell_1, \ell_2, \ell_3}^{\text{SD}_{\mathbf{B}_1 \rightarrow \mathbf{B}_2}^{\mathbb{G}_s}} = \left| \Pr [\mathcal{A}(\mathbb{G}, D, [\mathbf{t}_0^\top]_s) = 1] - \Pr [\mathcal{A}(\mathbb{G}, D, [\mathbf{t}_1^\top]_s) = 1] \right|$$

where $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, $D := ([\mathbf{B}_1]_s, [\mathbf{B}_2]_s, [\mathbf{B}_3]_s, \text{basis}(\mathbf{B}_1^\parallel, \mathbf{B}_2^\parallel), \text{basis}(\mathbf{B}_3^\parallel))$ and $\mathbf{t}_0 \leftarrow \text{span}(\mathbf{B}_1^\top)$, $\mathbf{t}_1 \leftarrow \text{span}(\mathbf{B}_2^\top)$

2.3 Registered Functional Encryption

Let \mathcal{F} be a function family such that, for all $f \in \mathcal{F}, f : \mathcal{X} \rightarrow \mathcal{Z}$. A Registered Functional Encryption (Reg-FE) for \mathcal{F} consists of six algorithms:

- $\text{Setup}(1^\lambda, 1^L, F) \rightarrow \text{crs}$: It takes a security parameter 1^λ , the maximum number of users 1^L , and a function family \mathcal{F} as input, and outputs a common reference string crs .
- $\text{Gen}(\text{crs}, \text{aux}) \rightarrow (\text{pk}, \text{sk})$: It takes crs and state aux as input, and outputs a key pair (pk, sk) .
- $\text{Reg}(\text{crs}, \text{aux}, \text{pk}, f) \rightarrow (\text{mpk}, \text{aux}')$: It takes crs , aux , a public key pk and a function $f \in \mathcal{F}$ as input, and outputs a master public key mpk and an updated state aux' .
- $\text{Upd}(\text{crs}, \text{aux}, \text{pk}) \rightarrow \text{hsk}$: It takes crs , aux , pk as input, and outputs a helper key hsk .
- $\text{Enc}(\text{mpk}, x) \rightarrow \text{ct}$: It takes mpk and $x \in X$ as input, and outputs a ciphertext ct .
- $\text{Dec}(\text{sk}, \text{hsk}, \text{ct}) \rightarrow z/\perp/\text{getupd}$: It takes $\text{sk}, \text{hsk}, \text{ct}$ as input, and outputs $z \in Z$, or a decryption failure symbol \perp , or a flag getupd prompting to update hsk .

Correctness, Compactness and Update Efficiency. A Reg-FE is correct if for all stateful adversary \mathcal{A} making a polynomial number of oracle queries and all L , the following advantage function is negligible in λ :

$$\Pr \left[b = 1 \mid \begin{array}{l} \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^L, F); b = 0; \\ \mathcal{A}^{\text{ORegNT}, \text{ORegT}, \text{OEnc}, \text{ODec}}(\text{crs}) \end{array} \right]$$

where oracles are defined as follows with $\text{aux} = \perp$, $\mathcal{E} \in \emptyset$, $R = \emptyset$, and $t = \perp$ at the beginning.

- $\text{ORegNT}(pk, f)$: It runs $(\text{mpk}, \text{aux}') \leftarrow \text{Reg}(\text{crs}, \text{pk}, f)$, updates $\text{aux} = \text{aux}'$, adds (mpk, aux) to R and returns $(|R|, \text{mpk}, \text{aux})$;
- $\text{ORegT}(f^*)$: It runs $(\text{pk}^*, \text{sk}^*) \leftarrow \text{Gen}(\text{crs}, \text{aux})$, $(\text{mpk}, \text{aux}') \leftarrow \text{Reg}(\text{crs}, \text{aux}, \text{pk}^*, f^*)$, updates $\text{aux} = \text{aux}'$, computes $\text{hsk}^* \leftarrow \text{Upd}(\text{crs}, \text{aux}, \text{pk}^*)$, appends (mpk, aux) to R , and returns $(t = |R|, \text{mpk}, \text{aux}, \text{pk}^*, \text{sk}^*, \text{hsk}^*)$;
- $\text{OEnc}(i, x)$: It sets $R[i] = (\text{mpk}, \star)$, runs $\text{ct} \leftarrow \text{Enc}(\text{mpk}, x)$, adds (x, ct) to \mathcal{E} and returns $(|\mathcal{E}|, \text{ct})$;
- $\text{ODec}(j)$: let $\mathcal{E}[j] = (x_j, \text{ct}_j)$, compute $z_j \leftarrow \text{Dec}(\text{sk}^*, \text{hsk}^*, \text{ct}_j)$; if $z_j = \text{getupd}$, run $\text{hsk}^* \leftarrow \text{Upd}(\text{crs}, \text{aux}, \text{pk}^*)$ and recompute $z_j \leftarrow \text{Dec}(\text{sk}^*, \text{hsk}^*, \text{ct}_j)$. Set $b = 1$ when $z_j \neq f^*(x_j)$.

with the following restrictions:

- there are at most $L - 1$ queries to ORegNT and there is exactly one query to ORegT ; therefore we will consider $f^*, \text{pk}^*, \text{sk}^*, \text{hsk}^*$ to be global;
- for query (i, x) to OEnc , it holds that $i \leq t$, $R[i] \neq \perp$;
- for query (j) to ODec , it holds that $E[j] \neq \perp$.

Adaptive IND-Security. For all stateful PPT adversary \mathcal{A} , the adaptive IND-security requires the advantage function $\text{Adv}_{\mathcal{A}}^{\text{adp-Reg-FE}}$ defined as follows is negligible in λ :

$$\begin{aligned} & \text{Adv}_{\mathcal{A}}^{\text{adp-Reg-FE}}(\lambda) \\ &= \Pr \left[b = b' \mid \begin{array}{l} \text{crs} \leftarrow \text{Setup}(1^\lambda, F); \\ x_0, x_1 \leftarrow \mathcal{A}^{\text{ORegCK, ORegHK, OCorHK}}(\text{crs}); \\ b \leftarrow \{0, 1\}, \text{ct}_b \leftarrow \text{Enc}(\text{mpk}, x_b), b' \leftarrow \mathcal{A}(\text{ct}_b) \end{array} \right] - 1/2, \end{aligned}$$

where the oracles are defined as follows with $\text{aux} = \perp, \text{mpk} = \perp, \mathcal{H} = \emptyset, \mathcal{C} = \emptyset$ and \mathcal{D} being a dictionary with $\mathcal{D}[\text{pk}] = \emptyset$ for all possible pk at the beginning of the game:

- $\text{ORegCK}(\text{pk}, f)$: $\text{run}(\text{mpk}', \text{aux}) \leftarrow \text{Reg}(\text{crs}, \text{aux}, \text{pk}, f)$, update $\text{mpk} = \text{mpk}'$, $\text{aux} = \text{aux}'$, $\mathcal{D}[\text{pk}] = \mathcal{D}[\text{pk}] \cup \{f\}$, append pk to \mathcal{C} and return (mpk, aux) ;
- $\text{ORegHK}(f)$: $\text{run}(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\text{crs}, \text{aux})$ and $(\text{mpk}', \text{aux}) \leftarrow \text{Reg}(\text{crs}, \text{aux}, \text{pk}, f)$, update $\text{mpk} = \text{mpk}'$ and $\text{aux} = \text{aux}'$, $\mathcal{D}[\text{pk}] = \mathcal{D}[\text{pk}] \cup \{f\}$, append (pk, sk) to \mathcal{H} and return $(|\mathcal{H}|, \text{mpk}, \text{aux}, \text{pk})$;
- $\text{OCorHK}(i)$: let $\mathcal{H}[i] = (\text{pk}, \text{sk})$, append pk to \mathcal{C} and return sk ;

with the following restrictions:

- for query i to OCorHK , it holds that $\mathcal{H}[i] \neq \perp$;
- for all $f \in \bigcup_{\text{pk} \in \mathcal{C}} \mathcal{D}[\text{pk}]$, it holds that $f(x_0) = f(x_1)$.

2.4 Slotted Registered Functional Encryption

For a modular approach to constructing Reg-FE schemes, the concept of slotted Reg-FE was introduced, which can be transformed to standard Reg-FE through a well-established “powers-of-two” technique, as demonstrated in [11,12,10,20]. A slotted Reg-FE consists of six algorithms, where Ver and Agg are deterministic algorithms:

- $\text{Setup}(1^\lambda, 1^L, \mathcal{F}) \rightarrow \text{crs}$: It takes as input security parameter 1^λ , maximum number of users 1^L , and a function family \mathcal{F} , and outputs a common reference string crs .
- $\text{Gen}(\text{crs}, i) \rightarrow (\text{pk}_i, \text{sk}_i)$: It takes as input crs and slot number $i \in [L]$, and outputs key pair $(\text{pk}_i, \text{sk}_i)$.
- $\text{Ver}(\text{crs}, i, \text{pk}_i) \rightarrow 0/1$: It takes as input crs, i and pk_i , and outputs a bit.
- $\text{Agg}(\text{crs}, (\text{pk}_i, f_i)_{i \in [L]}) \rightarrow (\text{mpk}, \{\text{hsk}_i\}_{i \in [L]})$: It takes as input crs , and public-key-function pairs $(\text{pk}_i, f_i)_{i \in [L]}$, and outputs master public key mpk and helper keys $\{\text{hsk}_i\}_{i \in [L]}$.
- $\text{Enc}(\text{mpk}, x) \rightarrow \text{ct}$: It takes as input mpk and $x \in \mathcal{X}$. It outputs a ciphertext ct .
- $\text{Dec}(\text{sk}_i, \text{hsk}_i, \text{ct}) \rightarrow z \in \mathcal{Z} / \perp$: It takes as input $\text{sk}_i, \text{hsk}_i$ and ct , and outputs $z \in \mathcal{Z}$ or a special symbol \perp to indicate a decryption failure.

Completeness. For all $\lambda, L \in \mathbb{N}$, all F , and all $i \in [L]$, we have

$$\Pr \left[\text{Ver}(\text{crs}, i, \text{pk}_i) = 1 \mid \begin{array}{l} \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^L, F); \\ (\text{pk}_i, \text{sk}_i) \leftarrow \text{Gen}(\text{crs}, i) \end{array} \right] = 1 - \text{negl}(\lambda).$$

Correctness. For all $\lambda, L \in \mathbb{N}$, all F , all $i^* \in [L]$, all $l \in \mathcal{L}ables$, all $\text{crs} \leftarrow \text{Setup}(1^\lambda, 1^L, \mathcal{F})$, all $(\text{pk}_{i^*}, \text{sk}_{i^*}) \leftarrow \text{Gen}(\text{crs}, i^*)$, all $\{\text{pk}_i\}_{i \in [L] \setminus \{i^*\}}$ such that $\text{Ver}(\text{crs}, i, \text{pk}_i) = 1$, all $x \in X$ and $f_1, \dots, f_L \in \mathcal{F}$, we have

$$\Pr \left[\text{Dec}(\text{sk}_{i^*}, \text{hsk}_{i^*}, \text{ct}) = f_{i^*}(x) \mid \begin{array}{l} (\text{mpk}, \{\text{hsk}_i\}_i) \leftarrow \text{Agg}(\text{crs}, \{\text{pk}_i, f_i\}_i) \\ \text{ct} \leftarrow \text{Enc}(\text{mpk}, x) \end{array} \right] = 1 - \text{negl}(\lambda).$$

Compactness. For all mpk and hsk_i , we have

$$|\text{mpk}| = \text{poly}(\lambda, P, \log L) \quad \text{and} \quad |\text{hsk}_i| = \text{poly}(\lambda, P, \log L).$$

where P is a parameter depending on the functionality F .

Adaptive-IND-Security. For all stateful PPT adversary \mathcal{A} , the adaptive IND-security requires the advantage function $\text{Adv}_{\mathcal{A}}^{\text{adp-sRFE}}$ defined as follows is negligible in λ :

$$\begin{aligned} & \text{Adv}_{\mathcal{A}}^{\text{adp-sRFE}}(\lambda) \\ &= \Pr \left[b = b' \mid \begin{array}{l} L \leftarrow \mathcal{A}(1^\lambda); \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^L, F) \\ \{\text{pk}_i^*, f_i^*\}_i, x_0, x_1 \leftarrow \mathcal{A}^{\text{OGen, OCor}}(\text{crs}) \\ (\text{mpk}, \{\text{hsk}_i\}) \leftarrow \text{Agg}(\text{crs}, \{\text{pk}_i^*, f_i^*\}_i) \\ b \leftarrow \{0, 1\}, \text{ct}_b \leftarrow \text{Enc}(\text{mpk}, x_b) \\ b' \leftarrow \mathcal{A}(\text{ct}_b) \end{array} \right] - \frac{1}{2} \end{aligned}$$

where the oracles work as follows with the initial setting $\mathcal{C} = \emptyset$ and $\mathcal{D}_i = \emptyset$ for all $i \in [L]$:

- $\text{OGen}(i)$: run $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\text{crs}, i)$, set $\mathcal{D}_i[\text{pk}] = \text{sk}$ and return pk .
- $\text{OCor}(i, \text{pk})$: return $\mathcal{D}_i[\text{pk}]$ and update $\mathcal{C} = \mathcal{C} \cup \{(i, \text{pk})\}$.

and for all $i \in [L]$, we require that

- $\text{Ver}(\text{crs}, i, \text{pk}_i^*) = 1$ for all i s.t. $\mathcal{D}_i[\text{pk}_i^*] = \perp$;
- $f_i^*(x_0^*) = f_i^*(x_1^*)$ for all i s.t. $(i, \text{pk}_i^*) \in \mathcal{C} \vee \mathcal{D}_i[\text{pk}_i^*] = \perp$.

Note that pk_i serves as a general entry in \mathcal{D}_i while pk_i^* is the specific challenge public for slot i ; there can be more than one assignments for pk_i since the adversary can query $\text{OGen}(i)$ for many times.

2.5 Tools

Quasi-Adaptive Non-Interactive Zero-Knowledge Argument. A Quasi-Adaptive Non-Interactive Zero-Knowledge (QA-NIZK) argument [14] for linear space over bilinear group \mathbb{G} from pairing consists of four efficient algorithms:

- $\text{LGen}(1^\lambda, 1^n, 1^m, 1^\ell, [\mathbf{A}]_1)$: It takes the security parameter 1^λ as input, language parameter $1^n, 1^m, 1^\ell$, and a matrix $[\mathbf{A}]_1 \leftarrow \mathbb{G}_1^{n \times m}$ defining a linear space, outputs a common reference string crs and trapdoor td .
- $\text{LPrv}(\text{crs}, [\mathbf{F}]_1, \mathbf{U})$: It takes crs as input, a matrix $[\mathbf{F}]_1 \in \mathbb{G}_1^{n \times \ell}$ along with $\mathbf{U} \in \mathbb{Z}_p^{m \times \ell}$ such that $\mathbf{F} = \mathbf{A}\mathbf{U}$, and outputs a proof π .
- $\text{LVer}(\text{crs}, [\mathbf{F}]_1, \pi)$: It takes $\text{crs}, [\mathbf{F}]_1$ and π as input, and outputs a bit showing the validity of π .
- $\text{LSim}(\text{crs}, \text{td}, [\mathbf{F}]_1)$: It takes $\text{crs}, \text{td}, [\mathbf{F}]_1$ as input, and outputs a simulation proof $\tilde{\pi}$.

Perfect Completeness. For all λ, \mathbf{A} , and all \mathbf{U}, \mathbf{F} such that $\mathbf{F} = \mathbf{A}\mathbf{U}$:

$$\Pr \left[\text{LVer}(\text{crs}, [\mathbf{F}]_1, \pi) = 1 \mid \begin{array}{l} (\text{crs}, \text{td}) \leftarrow \text{LGen}(1^\lambda, 1^n, 1^m, 1^\ell, [\mathbf{A}]_1) \\ \pi \leftarrow \text{LPrv}(\text{crs}, [\mathbf{F}]_1, \mathbf{U}) \end{array} \right] = 1.$$

Perfect Zero-knowledge. For all $\lambda, \mathbf{A}, (\text{crs}, \text{td}) \leftarrow \text{LGen}(1^\lambda, 1^n, 1^m, 1^\ell, [\mathbf{A}]_1)$, and all \mathbf{U}, \mathbf{F} such that $\mathbf{F} = \mathbf{A}\mathbf{U}$:

$$\text{LPrv}(\text{crs}, [\mathbf{F}]_1, \mathbf{U}) \equiv \text{LSim}(\text{crs}, \text{td}, [\mathbf{F}]_1).$$

Unbounded Simulation Soundness. For all adversary \mathcal{A} , the advantage

$$\Pr \left[\begin{array}{l} ([\mathbf{F}^*]_1, \pi) \notin \mathcal{Q} \wedge \\ \mathbf{F}^{*\top} \notin \text{span}(\mathbf{A}^\top) \wedge \\ \text{LVer}(\text{crs}, [\mathbf{F}^*]_1, \pi^*) = 1 \end{array} \mid \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_p^{n \times m} \\ (\text{crs}, \text{td}) \leftarrow \text{LGen}(1^\lambda, 1^n, 1^m, 1^\ell, [\mathbf{A}]_1) \\ ([\mathbf{F}^*]_1, \pi^*) \leftarrow \mathcal{A}^{\text{LSim}(\text{crs}, \text{td}, \cdot)}(1^\lambda, \text{crs}, \mathbf{A}) \end{array} \right]$$

is negligible in λ , where \mathcal{Q} records all queries to $\text{LSim}(\text{crs}, \text{td}, \cdot)$ along with response. Note that the definition is stronger in the sense that the adversary is given \mathbf{A} instead of $[\mathbf{A}]_1$.

Arithmetic Branching Programs (ABPs). An arithmetic branching program $f : \mathbb{Z}_p^{n_1} \rightarrow \mathbb{Z}_p$ is defined by a prime p , a directed acyclic graph (V, E) , two special vertices $v_0, v_1 \in V$ and a labeling function $\Phi : E \rightarrow \mathcal{F}^{\text{affine}}$ that assigns to each edge in E an affine function $g : \mathbb{Z}_p^{n_0} \rightarrow \mathbb{Z}_p$, and $f(x)$ is the sum over all $v_0 - v_1$ paths of the product of all the values along the path. We refer to $|V| + |E|$ as the size of f . The definition extends to functions $f : \mathbb{Z}_p^{n_1} \rightarrow \mathbb{Z}_p^{n_2}$ in a coordinate-wise manner. And we denote $\mathcal{F}_{n_1, n_2}^{\text{ABP}}$ the class of ABP $f : \mathbb{Z}_p^{n_1} \rightarrow \mathbb{Z}_p^{n_2}$.

Partial Garbling Scheme. A partial garbling scheme [13,18,2] for $f(\mathbf{x})^\top \mathbf{z}$ where $f \in \mathcal{F}_{\text{ABP}, n_1, n_2}$ consists four efficient algorithms $(\text{lgen}, \text{pgb}, \text{rec}, \text{pgb}^*)$, where lgen and rec are deterministic, with the following properties:

- (syntax) on input $f \in \mathcal{F}_{\text{ABP}, n_1, n'}$, $\text{lgen}(f)$ outputs $\mathbf{L}_0 \in \mathbb{Z}_p^{(m+n_2-1) \times mn_1}, \mathbf{L}_1 \in \mathbb{Z}_p^{(m+n_2-1) \times m}$, and

$$\begin{aligned} \text{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) &= (\mathbf{t}^\top (\mathbf{L}_1 (\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0), \mathbf{z}^\top - \bar{\mathbf{t}}^\top) \\ \text{pgb}^*(f, \mathbf{x}, \mu; \mathbf{t}) &= (\mathbf{t}^\top (\mathbf{L}_1 (\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0) + \mu \cdot \mathbf{e}_1^\top, \bar{\mathbf{t}}^\top) \end{aligned}$$

where $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n_2-1}$ and $\bar{\mathbf{t}}$ consists of the last n_2 entries in \mathbf{t} and m are linear in the size of f .

– (reconstruction) $\text{rec}(f, \mathbf{x})$ outputs $\mathbf{g}_{f, \mathbf{x}} \in \mathbb{Z}_p^{n_2+m}$ such that for all $f, \mathbf{x}, \mathbf{z}, \mathbf{t}$,

$$\text{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) \mathbf{g}_{f, \mathbf{x}}^\top = f(\mathbf{x})^\top \mathbf{z}.$$

– (privacy) for all $f, \mathbf{x}, \mathbf{z}$,

$$\text{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) \approx_s \text{pgb}^*(f, \mathbf{x}, f(\mathbf{x})^\top \mathbf{z}; \mathbf{t})$$

where the randomness is over $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n_2-1}$.

Extension. We will also rely on an extra property of the above construction to handle shifts by $\delta \in \mathbb{Z}_p$ and define a new algorithm pgb^+ .

$$\text{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}) = (\mathbf{t}^\top (\mathbf{L}_1 (\mathbf{x} \otimes \mathbf{I}_m) + \mathbf{L}_0) + \delta \cdot \mathbf{e}_1^\top, \mathbf{z}^\top - \bar{\mathbf{t}}^\top)$$

together with (f, \mathbf{x}) , we can recover $f(\mathbf{x})^\top \mathbf{z} + \delta$, while learning nothing else about \mathbf{z}, δ . That is, for all $f, \mathbf{x}, \mathbf{z}$ and $\delta \in \mathbb{Z}_p$:

– (reconstruction)

$$\text{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}) = (\text{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) + (\delta \cdot \mathbf{e}_1^\top \|\mathbf{0}\rangle)) \mathbf{g}_{f, \mathbf{x}}^\top = f(\mathbf{x})^\top \mathbf{z} + \delta$$

– (privacy)

$$\text{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}) = \text{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) + (\delta \cdot \mathbf{e}_1^\top \|\mathbf{0}\rangle) \approx_s \text{pgb}^*(f, \mathbf{x}, f(\mathbf{x})^\top \mathbf{z} + \delta; \mathbf{t})$$

where the randomness is over $\mathbf{t} \leftarrow \mathbb{Z}_p^{m+n_2-1}$.

3 Slotted MultiReg-FE for Inner-Product

In this section, we introduce MultiReg-FE and give the construction. First of all, we show the definition of inner-product functionality.

Definition 3 (Inner-Product Functionality). *This definition is adapted from that in [1]. The function family $\mathcal{F}_{n,m}^{IP}$ consists of functions $f_{\mathbf{y}_1, \dots, \mathbf{y}_m} : (R^n)^m \rightarrow R$ where R is either \mathbb{Z} or \mathbb{Z}_p for some integer p , $\mathbf{y}_i \in R^n$ for $i \in [m]$ and*

$$f_{\mathbf{y}_1, \dots, \mathbf{y}_m}(\mathbf{x}_1, \dots, \mathbf{x}_m) = \sum_{i=1}^m \langle \mathbf{x}_i, \mathbf{y}_i \rangle,$$

and the vectors satisfy the following bounds: $\|\mathbf{x}_i\|_\infty < X$, $\|\mathbf{y}_i\|_\infty < Y$, for $i \in [m]$. X and Y are some positive integer. If omitted, then $X = Y = p$ is used (i.e. no constraint).

Remark 1. For single-input inner-product functionality, $m = 1$. For multi-input inner-product functionality, $m > 1$.

Remark 2. This definition applies to bounded inner product functionality, while for unbounded one, we will elaborate it in Definition 4.

The above is a common definition of the inner-product functionality. We give a more specific definition of Inner-Product in MultiReg-FE that is close to the multi-input one.

Inner-Product Functionality in MultiReg-FE. For ease of understanding, we can conceptually view inner-product functionality in MultiReg-FE as a new functionality $\mathcal{F}_n^{\text{multi-IP}}$ which is a variant of inner-product, thus MultiReg-FE is a case of Reg-FE. This function family $\mathcal{F}_n^{\text{multi-IP}}$ consists of functions $f_{\{\mathbf{y}_i\}_{i \in [m]}} : (R^n)^* \times \mathcal{L} \rightarrow R$ where $m \in \mathbb{N}$ and m is polynomial, $\mathbf{y}_i \in R^n$ for $i \in [m]$, R is either \mathbb{Z} or \mathbb{Z}_p for some integer p , and \mathcal{L} is a family of labeling functions $\phi(\cdot) : \mathbb{N} \rightarrow \mathbb{N}$. We defines for any $N \in \mathbb{N}$ and labeling function $\phi(\cdot) \in \mathcal{L}$ s.t. $\phi(N) \subseteq [m]$, for any input $\mathbf{x}_1, \dots, \mathbf{x}_N$, the function

$$f_{\{\mathbf{y}_i\}_{i \in [m]}}(\mathbf{x}_1, \dots, \mathbf{x}_N, \phi(\cdot)) = \sum_{j=1}^N \langle \mathbf{x}_j, \mathbf{y}_{\phi(j)} \rangle.$$

The vectors satisfy the same bounds as in inner-product functionality.

Note that the summation is over the subscript j . And we don't limit the number of vectors $\{\mathbf{x}_i\}^*$ and $\{\mathbf{y}_i\}^*$.

Slotted MultiReg-FE for Inner-Product. Having defined the inner-product functionality, we now proceed to introduce the definition of MultiReg-FE for IP. It is important to note that we treat MultiReg-FE for IP as a specific case of a Reg-FE scheme, but one that corresponds to the new functionality family $\mathcal{F}_n^{\text{multi-IP}}$. Therefore, we highlight the differences between MultiReg-FE for IP and the general Reg-FE for IP as follows:

- Agg $\left(\text{crs}, (\text{pk}_i, \{\mathbf{y}_{i,r}\}_{r \in [m_i]})_{i \in [L]} \right) \rightarrow (\text{mpk}, \{\text{hsk}_i\}_{i \in [L]})$: It takes as input crs, and public-key-function pairs $(\text{pk}_i, \{\mathbf{y}_{i,r}\}_{r \in [m_i]})_{i \in [L]}$, and outputs master public key mpk and helper keys $\{\text{hsk}_i\}_{i \in [L]}$.
- Enc $(\text{mpk}, \{\mathbf{x}_j\}_{j \in [N]}, \phi) \rightarrow \text{ct}$: It takes as input mpk, $\{\mathbf{x}_j\}_{j \in [N]}$ and a labeling function ϕ . It outputs a ciphertext ct.

In the next session, we will show the construction of MultiReg-FE for IP.

3.1 Construction

Let $\Pi_0 = (\text{LGen}, \text{LPrv}, \text{LVer}, \text{LSim})$ be a QA-NIZK scheme for linear space over bilinear groups.

Setup $(1^\lambda, 1^L, \mathcal{F}_n^{\text{multi-IP}})$:

It runs $\mathbb{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ and samples

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}, \mathbf{B}_1 \leftarrow \mathbb{Z}_p^{(2k+1) \times k}$$

For all $i \in [L]$, it samples

$$\mathbf{W}_i \leftarrow \mathbb{Z}_p^{(2k+1) \times (n+1)(2k+1)}, \mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+1)}, \mathbf{r}_i \leftarrow \mathbb{Z}_p^{1 \times k}.$$

For all $i \in [L]$, it writes $\mathbf{A}_i = \begin{pmatrix} \mathbf{A} \\ \mathbf{R}_i \end{pmatrix} \in \mathbb{Z}_p^{(3k+2) \times (2k+1)}$, runs

$$(\text{crs}_i, \text{td}_i) \leftarrow \text{LGen}(1^\lambda, 1^{3k+2}, 1^{2k+1}, 1^{2k+1}, [\mathbf{A}_i]_1)$$

and outputs⁴

$$\text{crs} = \left(\begin{array}{l} [\mathbf{A}]_1, \{[\mathbf{B}_1 \mathbf{r}_i^\top]_2\}_{i \in [L]} \\ \{\text{crs}_i, [\mathbf{R}_i, \mathbf{A} \mathbf{W}_i]_1\}_{i \in [L]} \\ \{[\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2\}_{i \in [L], u \in [L] \setminus \{i\}} \end{array} \right)$$

Gen(crs, i):

It chooses $\mathbf{U}_i \leftarrow \mathbb{Z}_p^{(2k+1) \times (2k+1)}$. It defines $\mathbf{F}_i = \begin{pmatrix} \mathbf{T}_i \\ \mathbf{Q}_i \end{pmatrix} = \begin{pmatrix} \mathbf{A} \mathbf{U}_i \\ \mathbf{R}_i \mathbf{U}_i \end{pmatrix} = \mathbf{A}_i \mathbf{U}_i \in \mathbb{Z}_p^{(3k+2) \times (2k+1)}$, runs

$$\pi_i \leftarrow \text{LPrv}(\text{crs}_i, [\mathbf{F}_i]_1, \mathbf{U}_i),$$

and outputs

$$\begin{aligned} \text{pk}_i &= (\mathbf{T}_i, \mathbf{Q}_i, \{\mathbf{h}_{i,u}\}_{u \in [L] \setminus \{i\}}, \pi_i) = ([\mathbf{A} \mathbf{U}_i, \mathbf{R}_i \mathbf{U}_i]_1, \{[\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus \{i\}}, \pi_i) \\ \text{sk}_i &= \mathbf{U}_i \end{aligned}$$

Ver(crs, i , pk_i):

It parses $\text{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{\mathbf{h}_{i,u}\}_{u \in [L] \setminus \{i\}}, \pi_i)$, sets $\mathbf{F}_i = \begin{pmatrix} \mathbf{T}_i \\ \mathbf{Q}_i \end{pmatrix}$ and checks

$$\text{LVer}(\text{crs}_i, [\mathbf{F}_i]_1, \pi_i) \stackrel{?}{=} 1$$

For each $u \in [L] \setminus \{i\}$, it checks

$$e([\mathbf{A}]_1, [\mathbf{h}_{i,u}]_2) \stackrel{?}{=} e([\mathbf{T}_i]_1, [\mathbf{B}_1 \mathbf{r}_u^\top]_2).$$

If all these checks pass, it outputs 1; otherwise, it outputs 0.

Agg(crs, $(\text{pk}_i, \{\mathbf{y}_{i,r}\}_{r \in [m_i]})_{i \in [L]}$):

For all $i \in [L]$, it takes crs , $\text{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{\mathbf{h}_{i,u}\}_{u \in [L] \setminus \{i\}}, \pi_i)$ and $\mathbf{y}_{i,1}, \dots,$

⁴ Note that $\text{td}_1, \dots, \text{td}_L$ are not used in the actual scheme and $\{\mathbf{R}_i\}_i$ are only actually used in the verification algorithm Ver . However, both will be used in the security proof.

$\mathbf{y}_{i,m_i} \in \mathbb{Z}_p^n$ as input, where $m_i = \text{poly}(\lambda)$. It defines $M = \max_{i \in [L]} m_i$ and for $r \in [M]$ sets

$$\mathbf{y}'_{i,r} = \begin{cases} (\mathbf{y}_{i,r}, 1) & \text{if } r \leq m_i \\ 0^{n+1} & \text{if } r > m_i \end{cases}.$$

It outputs $\text{mpk} = [\mathbf{P}_0, \{\mathbf{P}_{1,r}\}_{r \in [M]}, \mathbf{P}_2]_1$, and $\text{hsk}_i = [\mathbf{k}_{i,0}^\top, \{\mathbf{k}_{i,1,r}^\top\}_{r \in [m_i]}, \mathbf{K}_{i,2}]_2$ for all $i \in [L]$, where

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{A} \\ \mathbf{P}_{1,r} &= \sum_{u \in [L]} (\mathbf{T}_u + \mathbf{A}\mathbf{W}_u (\mathbf{y}'_{u,r}{}^\top \otimes \mathbf{I}_{2k+1})) \\ \mathbf{P}_2 &= \sum_{u \in [L]} \mathbf{A}\mathbf{W}_u \\ \mathbf{k}_{i,0}^\top &= \mathbf{B}_1 \mathbf{r}_i^\top \\ \mathbf{k}_{i,1,r}^\top &= \sum_{u \in [L] \setminus \{i\}} (\mathbf{h}_{u,i} + \mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{u,r}{}^\top) \\ \mathbf{K}_{i,2} &= \sum_{u \in [L] \setminus \{i\}} \mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \end{aligned}$$

Enc(mpk, $\{\mathbf{x}_j\}_{j \in [N]}, \phi$):

It chooses $\mathbf{v} \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, $\mathbf{s}_1, \dots, \mathbf{s}_N \leftarrow \mathbb{Z}_p^{1 \times k}$, $\rho_1, \dots, \rho_N \leftarrow \mathbb{Z}_p$ s.t. $\sum_{j \in [N]} \rho_j = 0$,

sets

$$\mathbf{x}'_j = (\mathbf{x}_j, \rho_j)$$

and computes

$$\begin{aligned} \mathbf{c}_{0,j} &= \mathbf{s}_j \mathbf{P}_0 &= \mathbf{s}_j \mathbf{A} \\ \mathbf{c}_{1,j} &= \mathbf{s}_j \mathbf{P}_{1,\phi(j)} &= \sum_{u \in [L]} (\mathbf{s}_j \mathbf{T}_u + \mathbf{s}_j \mathbf{A}\mathbf{W}_u (\mathbf{y}'_{u,\phi(j)}{}^\top \otimes \mathbf{I}_{2k+1})) \\ \mathbf{c}_{2,j} &= \mathbf{x}'_j \otimes \mathbf{v} + \mathbf{s}_j \mathbf{P}_2 &= \mathbf{x}'_j \otimes \mathbf{v} + \mathbf{s}_j \sum_{u \in [L]} \mathbf{A}\mathbf{W}_u \\ \mathbf{c}_3 &= \mathbf{v} \end{aligned}$$

It outputs $\text{ct} = ([\{\mathbf{c}_{0,j}, \mathbf{c}_{1,j}, \mathbf{c}_{2,j}\}_{j \in [N]}, \mathbf{c}_3]_1, \phi)$.

Dec(sk_i, hsk_i, ct):

It parses

$$\text{sk}_i = \mathbf{U}_i, \quad \text{hsk}_i = [\mathbf{k}_{i,0}^\top, \{\mathbf{k}_{i,1,r}^\top\}_{r \in [m_i]}, \mathbf{K}_{i,2}]_2, \quad \text{ct} = [\{\mathbf{c}_{0,j}, \mathbf{c}_{1,j}, \mathbf{c}_{2,j}\}_{j \in [N]}, \mathbf{c}_3]_1,$$

and for all $j \in [N]$, computes

$$\begin{aligned} [\mathbf{z}_{1,j}]_T &= e([\mathbf{c}_{2,j}]_1, [\mathbf{I}_n \otimes \mathbf{k}_{i,0}^\top]_2), & [\mathbf{z}_{2,j}]_T &= e([\mathbf{c}_{0,j}]_1, [\mathbf{K}_{i,2}]_2), \\ [\mathbf{z}_{3,j}]_T &= e([\mathbf{c}_{1,j}]_1, [\mathbf{k}_{i,0}^\top]_2), & [\mathbf{z}_{4,j}]_T &= e([\mathbf{c}_{0,j}]_1, [\mathbf{k}_{i,1,\phi(j)}^\top]_2), \\ [\mathbf{z}_{5,j}]_T &= e([\mathbf{c}_{0,j} \mathbf{U}_i]_1, [\mathbf{k}_{i,0}^\top]_2), \\ [\mathbf{z}_6]_T &= e([\mathbf{c}_3]_1, [\mathbf{k}_{i,0}^\top]_2), \\ [\mathbf{z}'_j]_T &= [(z_{1,j} - z_{2,j}) \mathbf{y}'_{i,\phi(j)}{}^\top - (z_{3,j} - z_{4,j} - z_{5,j})]_T \\ [\mathbf{z}']_T &= \left[\sum_{j \in [N]} \mathbf{z}'_j \right]_T. \end{aligned}$$

It recovers z from $[z']_T$ over $[z_6]_T$ via brute-force DLog and outputs $z = \sum_{j \in [N]} z_j$.

Completeness. For all $\lambda, L, n \in \mathbb{N}$, all $i \in [L]$, all $\text{crs} \leftarrow \text{Setup}(1^\lambda, 1^n, 1^L, \mathcal{F}^{\text{multi-IP}})$ and $(\text{pk}_i, \text{sk}_i) \leftarrow \text{Gen}(\text{crs}, i)$, we have

$$\begin{aligned} \text{pk}_i &= \left([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{h}_{i,u}]_2\}_{u \in [L] \setminus \{i\}}, \pi_i \right) \\ &= \left([\mathbf{A}\mathbf{U}_i, \mathbf{R}_i\mathbf{U}_i]_1, \{[\mathbf{U}_i\mathbf{B}_1\mathbf{r}_u^\top]_2\}_{u \in [L] \setminus \{i\}}, \pi_i \right) \end{aligned}$$

for some $\mathbf{U}_i \leftarrow \mathbb{Z}_p^{(2k+1) \times (2k+1)}$ and $\pi_i \leftarrow \text{LPrv}(\text{crs}_i, [\mathbf{A}_i\mathbf{U}_i]_1, \mathbf{U}_i)$ where $(\text{crs}_i, \text{td}_i) \leftarrow \text{LGen}(1^\lambda, 1^{3k+2}, 1^{2k+1}, 1^{2k+1}, [\mathbf{A}_i]_1)$ and $\mathbf{A}_i = \begin{pmatrix} \mathbf{A} \\ \mathbf{R}_i \end{pmatrix}$ with $\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}$, $\mathbf{R}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+1)}$. We have $\text{LVer}(\text{crs}_i, [\mathbf{F}_i]_1, \pi_i) = 1$ by the perfect completeness of Π_0 and the fact that $\mathbf{F}_i = \mathbf{A}_i\mathbf{U}_i$; For each $u \in [L] \setminus \{i\}$, we have $e([\mathbf{A}]_1, [\mathbf{U}_i\mathbf{B}_1\mathbf{r}_u^\top]_2) = e([\mathbf{A}\mathbf{U}_i]_1, [\mathbf{B}_1\mathbf{r}_u^\top]_2)$ from the fact that $\mathbf{A}(\mathbf{U}_i\mathbf{B}_1\mathbf{r}_u^\top) = (\mathbf{A}\mathbf{U}_i)(\mathbf{B}_1\mathbf{r}_u^\top)$. Thus, $\text{Ver}(\text{crs}, i, \text{pk}_i) = 1$.

Correctness. In decryption, we have

$$\begin{aligned}
[z_{1,j}]_T &= e\left([\mathbf{c}_{2,j}]_1, [\mathbf{I}_n \otimes \mathbf{k}_{i,0}^\top]_2\right) \\
&= [(\mathbf{x}'_j \mathbf{I}_n) \otimes (\mathbf{v} \mathbf{B}_1 \mathbf{r}_i^\top) + \sum_{u \in [L]} \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_T \\
&= [\mathbf{v} \mathbf{B}_1 \mathbf{r}_i^\top \mathbf{x}'_j + \sum_{u \in [L]} \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_T \\
[z_{2,j}]_T &= e([\mathbf{c}_{0,j}]_1, [\mathbf{K}_{i,2}]_2) \\
&= [\mathbf{s}_j \mathbf{A} \sum_{u \in [L] \setminus \{i\}} (\mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top))]_T \\
&= [\sum_{u \in [L] \setminus \{i\}} \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_T \\
[z_{3,j}]_T &= e([\mathbf{c}_{1,j}]_1, [\mathbf{k}_{i,0}^\top]_2) \\
&= [\sum_{u \in [L]} (\mathbf{s}_j \mathbf{T}_u + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)} \otimes \mathbf{I}_{2k+1})) \mathbf{B}_1 \mathbf{r}_i^\top]_T \\
&= [\sum_{u \in [L]} (\mathbf{s}_j \mathbf{T}_u \mathbf{B}_1 \mathbf{r}_i^\top + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)} \otimes \mathbf{I}_{2k+1}) \mathbf{B}_1 \mathbf{r}_i^\top)]_T \\
&= [\sum_{u \in [L]} (\mathbf{s}_j \mathbf{T}_u \mathbf{B}_1 \mathbf{r}_i^\top + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{u,\phi(j)})]_T \\
[z_{4,j}]_T &= e([\mathbf{c}_{0,j}]_1, [\mathbf{k}_{i,1,j}^\top]_2) \\
&= [\mathbf{s}_j \mathbf{A} \sum_{u \in [L] \setminus \{i\}} (\mathbf{h}_{u,i} + \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{u,\phi(j)})]_T \\
&= [\sum_{u \in [L] \setminus \{i\}} (\mathbf{s}_j \mathbf{A} \mathbf{h}_{u,i} + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{u,\phi(j)})]_T \\
[z_{5,j}]_T &= e([\mathbf{c}_{0,j} \mathbf{U}_i]_1, [\mathbf{k}_{i,0}^\top]_2) \\
&= [\mathbf{s}_j \mathbf{A} \mathbf{U}_i \mathbf{B}_1 \mathbf{r}_i^\top]_T \\
[z_6]_T &= e([\mathbf{c}_3]_1, [\mathbf{k}_{i,0}^\top]_2) \\
&= [\mathbf{v} \mathbf{B}_1 \mathbf{r}_i^\top]_T
\end{aligned}$$

Due to $\mathbf{A} \mathbf{h}_{u,i} = \mathbf{T}_u \mathbf{B}_1 \mathbf{r}_i^\top$ for $u \neq i$, and $\mathbf{A} \mathbf{U}_i = \mathbf{T}_i$, we have

$$\begin{aligned}
[z_{4,j}]_T &= [\sum_{u \in [L] \setminus \{i\}} \mathbf{s}_j \mathbf{T}_u \mathbf{B}_1 \mathbf{r}_i^\top + \sum_{u \in [L] \setminus \{i\}} \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{u,\phi(j)}]_T \\
[z_{5,j}]_T &= [(\mathbf{s}_j \mathbf{T}_i \mathbf{B}_1 \mathbf{r}_i^\top)]_T \\
[z_{3,j} - z_{4,j} - z_{5,j}]_T &= \mathbf{s}_j \mathbf{A} \mathbf{W}_i (\mathbf{I}_n \otimes \mathbf{B}_1 \mathbf{r}_i^\top) \mathbf{y}'_{i,\phi(j)}
\end{aligned}$$

Then we have

$$\begin{aligned}
[z_{1,j} - z_{2,j}]_T &= \mathbf{v}\mathbf{B}_1\mathbf{r}_i^\top \mathbf{x}'_j + \mathbf{s}_j\mathbf{A}\mathbf{W}_i(\mathbf{I}_n \otimes \mathbf{B}_1\mathbf{r}_i^\top) \\
[z'_j]_T &= [(\mathbf{z}_{1,j} - \mathbf{z}_{2,j})\mathbf{y}'_{i,\phi(j)} - (z_{3,j} - z_{4,j} - z_{5,j})]_T \\
&= \mathbf{v}\mathbf{B}_1\mathbf{r}_i^\top \mathbf{x}'_j \mathbf{y}'_{i,\phi(j)\top} \\
[z']_T &= [\sum_{j \in [N]} z'_j]_T \\
&= [\mathbf{v}\mathbf{B}_1\mathbf{r}_i^\top \sum_{j \in [N]} \langle \mathbf{x}'_j, \mathbf{y}'_{i,\phi(j)} \rangle]_T \\
&= [\mathbf{v}\mathbf{B}_1\mathbf{r}_i^\top \sum_{j \in [N]} \langle (\mathbf{x}_j, \rho_j), (\mathbf{y}_{i,\phi(j)}, 1) \rangle] \\
&= [\mathbf{v}\mathbf{B}_1\mathbf{r}_i^\top \sum_{j \in [N]} \langle \mathbf{x}_j, \mathbf{y}_{i,\phi(j)} \rangle]
\end{aligned}$$

Finally, solve the discrete logarithm in basis $[z_6]_T$ to extract $\sum_{j \in [N]} \langle \mathbf{x}_j, \mathbf{y}_{i,\phi(j)} \rangle$ via brute-force DLog.

Compactness and Efficiency. Our slotted MultiReg-FE has the following properties:

$$\begin{aligned}
|\text{crs}| &= L^2 \cdot (n+1) \cdot \text{poly}(\lambda); & |\text{mpk}| &= (n+1) \cdot M \cdot \text{poly}(\lambda); \\
|\text{hsk}_i| &= (n+1) \cdot m_i \cdot \text{poly}(\lambda); & |\text{ct}| &= (n+1) \cdot N \cdot \text{poly}(\lambda).
\end{aligned}$$

3.2 Security

Theorem 1. *If QA-NIZK $\Pi_0 = (\text{LGen}, \text{LPrv}, \text{LVer}, \text{LSim})$ has perfect completeness, perfect zero-knowledge and unbounded simulation soundness for linear space, and the MDDH and SD assumptions hold, our slotted MultiReg-FE scheme achieves the adaptive IND-security.*

Proof.

We prove the Theorem 1 via nested dual-system method. The series of game sequence is as follows.

Game Sequence. Suppose that crs is the common reference string, $(\{\mathbf{x}_{0,j}\}_{j \in [N]}, \{\mathbf{x}_{1,j}\}_{j \in [N]})$ is the challenge pair, $\{\text{pk}_i^*, \{\mathbf{y}_{i,r}\}_{r \in [m_i]}\}_{i \in [L]}$ are challenge public keys along with challenge functions to be registered. For all $i \in [L]$, define $D_i = \{\text{pk}_i : \mathcal{D}_i[\text{pk}_i] \neq \perp\}$ as responses from $\text{OGen}(i)$ and $C_i = \{\text{pk}_i : (i, \text{pk}_i) \in C\}$ as public keys in D_i queried to $\text{OCor}(i, \cdot)$. For all $i \in [L]$, we require that

- $\text{Ver}(\text{crs}, i, \text{pk}_i^*) = 1$ for all i s.t. $\mathcal{D}_i[\text{pk}_i^*] = \perp$;
- $\sum_{j \in [N]} \langle \mathbf{x}_{0,j}, \mathbf{y}_{i,\phi(j)} \rangle = \sum_{j \in [N]} \langle \mathbf{x}_{1,j}, \mathbf{y}_{i,\phi(j)} \rangle$ for all i s.t. $((i, \text{pk}_i^*) \in C \vee \mathcal{D}_i[\text{pk}_i^*] = \perp) \wedge \phi([N]) \subseteq [m_{i^*}]$.

Note that there can be more than one assignments for pk_i since the adversary can query $\text{OGen}(i)$ for many times.

- G_0 : This is the real game. Specifically,
 - crs is in the form:

$$\text{crs} = \left(\begin{array}{l} [\mathbf{A}]_1, \{ [\mathbf{B}_1 \mathbf{r}_i^\top]_2 \}_{i \in [L]} \\ \{ \text{crs}_i, [\mathbf{R}_i, \mathbf{A} \mathbf{W}_i]_1 \}_{i \in [L]} \\ \{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2 \}_{i \in [L], u \in [L] \setminus \{i\}} \end{array} \right)$$

where $\text{crs}_i \leftarrow \text{LGen}(1^\lambda, 1^{3k+2}, 1^{2k+1}, 1^{2k+1}, [\mathbf{A}_i]_1)$, $\mathbf{A}_i = \begin{pmatrix} \mathbf{A} \\ \mathbf{R}_i \mathbf{U}_i \end{pmatrix}$.

- For each $i \in [L]$, each pair $(\text{pk}_i, \text{sk}_i) \in D_i$ is in the form:

$$\begin{aligned} \text{pk}_i &= (\mathbf{T}_i, \mathbf{Q}_i, \{ \mathbf{h}_{i,j} \}_{j \in [L] \setminus \{i\}}, \pi_i) \\ &= ([\mathbf{A} \mathbf{U}_i, \mathbf{R}_i \mathbf{U}_i]_1, \{ [\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_j^\top]_2 \}_{j \in [L] \setminus \{i\}}, \pi_i) \\ \text{sk}_i &= \mathbf{U}_i \end{aligned}$$

where $\pi_i \leftarrow \text{LPrv}(\text{crs}_i, [\mathbf{F}_i]_1, \mathbf{U}_i)$, $\mathbf{F}_i = \begin{pmatrix} \mathbf{A} \mathbf{U}_i \\ \mathbf{R}_i \mathbf{U}_i \end{pmatrix}$.

- For all $i \in [L]$, pk_i^* is in the form:

$$\text{pk}_i^* = \left([\mathbf{T}_i^*, \mathbf{Q}_i^*]_1, \{ [\mathbf{h}_{i,u}^*]_2 \}_{u \in [L] \setminus \{i\}}, \pi_i^* \right)$$

such that $\text{Ver}(\text{crs}, i, \text{pk}_i^*) = 1$, which means $\text{LVer} \left(\text{crs}_i, \begin{bmatrix} \mathbf{T}_i^* \\ \mathbf{Q}_i^* \end{bmatrix}_1, \pi_i^* \right) = 1$

and $\mathbf{A} \mathbf{h}_{i,u}^* = \mathbf{T}_i^* \mathbf{B}_1 \mathbf{r}_u^\top$ for each $u \in [L] \setminus \{i\}$.

- ct for $(\mathbf{x}_0, \mathbf{x}_1)$ is in the form:

$$\begin{aligned} \text{ct} &= \left[\{ \mathbf{c}_{0,j} \}_{j \in [N]}, \{ \mathbf{c}_{1,j} \}_{j \in [N]}, \{ \mathbf{c}_{2,j} \}_{j \in [N]}, \mathbf{c}_3 \right]_1 \\ &= \left[\begin{array}{l} \{ \mathbf{s}_j \mathbf{A} \}_{j \in [N]}, \\ \left\{ \sum_{u \in [L]} \left(\mathbf{s}_j \mathbf{T}_u + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)}^\top \otimes \mathbf{I}_{2k+1}) \right) \right\}_{j \in [N]}, \\ \left\{ \sum_{u \in [L]} \left(\mathbf{s}_j \mathbf{T}_u + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)}^\top \otimes \mathbf{I}_{2k+1}) \right) \right\}_{j \in [N]}, \\ \{ \mathbf{x}'_{b,j} \otimes \mathbf{v} + \sum_{u \in [L]} \mathbf{s}_j \mathbf{A} \mathbf{W}_u \}_{j \in [N]}, \\ \mathbf{v} \end{array} \right]_1 \end{aligned}$$

where $b \leftarrow \{0, 1\}$ is the secret bit, $\mathbf{x}'_{b,j} = (\mathbf{x}_{b,j}, \rho_{b,j})$ and $\sum_{j \in [N]} \rho_{b,j} = 0$.

- G_1 : Same as G_0 , except that for all $i \in [L]$ and all $\text{pk}_i \in D_i$, we replace π_i with

$$\boxed{\tilde{\pi}_i \leftarrow \text{LSim}(\text{crs}_i, \text{td}_i, [\mathbf{F}_i]_1)} \quad \text{where} \quad \mathbf{F}_i = \begin{pmatrix} \mathbf{A} \mathbf{U}_i \\ \mathbf{R}_i \mathbf{U}_i \end{pmatrix}$$

We have $G_1 \equiv G_0$ from the perfect zero-knowledge of Π_0 .

- G_2 : Same as G_1 , except that we view all \mathbf{R}_i in crs as

$$\widehat{\mathbf{R}}_i = \begin{pmatrix} \widetilde{\mathbf{R}}_i \\ \mathbf{t}_i \end{pmatrix}, \quad \widetilde{\mathbf{R}}_i \leftarrow \mathbb{Z}_p^{(2k+2) \times (2k+2)}, \quad \mathbf{t}_i \leftarrow \mathbb{Z}_p^{1 \times (2k+2)}$$

We have $G_2 \equiv G_1$ from the fact that \mathbf{R}_i and $\widehat{\mathbf{R}}_i$ have the same randomness.

- G_3 : Same as G_2 , except that we replace all $\mathbf{c}_{2,j}$ as follows:

$$\mathbf{c}_{1,j} = \sum_{u \in [L]} \left(\mathbf{s}_j \mathbf{A} \widetilde{\mathbf{R}}_u^{-1} (\mathbf{I}_{2k+1} \parallel \mathbf{0}^\top) \mathbf{Q}_u^* + \mathbf{s}_j \mathbf{A} \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)}{}^\top \otimes \mathbf{I}_{2k+1}) \right)$$

We have $G_3 \approx_c G_2$. This follows from stronger unbounded simulation soundness of Π_0 along with the fact that $\text{LVer}(\text{crs}_i, [\mathbf{F}_i^*], \pi_i^*) = 1$ for all $i \in [L]$

where $\mathbf{F}_i^* = \begin{pmatrix} \mathbf{T}_i^* \\ \mathbf{Q}_i^* \end{pmatrix}$. Assume $\text{pk}_{i^*}^* \notin D_{i^*}$, i.e., $\text{pk}_{i^*}^*$ is malicious. In the reduction, we guess $i^* \leftarrow [L]$ and obtain $\mathbf{A}, \widehat{\mathbf{R}}_{i^*}, \text{crs}_{i^*}$ as input; we simulate honestly as in G_3 except that for all $\text{pk}_{i^*}^* \in D_{i^*}$, we make an oracle query $[\mathbf{F}_{i^*}^*]_1$ and get $\widetilde{\pi}_{i^*}^*$ in it; we finally output $([\mathbf{F}_{i^*}^*]_1, \pi_{i^*}^*)$ in $\text{pk}_{i^*}^* \notin D_{i^*}$. Observe that once it happens that $\mathbf{s}_j \mathbf{A} \widehat{\mathbf{R}}_{i^*}^{-1} (\mathbf{I}_{2k+1} \parallel \mathbf{0}^\top) \mathbf{Q}_i^* \neq \mathbf{s}_j \mathbf{T}_{i^*}^*$, we must have $\mathbf{F}_{i^*}^* \notin \text{span}(\mathbf{A}_{i^*})$, which happens with negligible probability because of stronger unbounded simulation soundness. And when $\text{pk}_{i^*}^* \in D_{i^*}$, we always have $G_3 \equiv G_2$.

- G_4 : Same as G_3 except that we replace all $\mathbf{s}_j \mathbf{A}$ with $\mathbf{a}_j \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, that is the challenge ciphertext

$$\begin{aligned} \text{ct} &= \{ \{\mathbf{c}_{0,j}\}_{j \in [N]}, \{\mathbf{c}_{1,j}\}_{j \in [N]}, \{\mathbf{c}_{2,j}\}_{j \in [N]}, \mathbf{c}_3 \}_1 \\ &= \left[\begin{array}{l} \{\mathbf{a}_j\}_{j \in [N]}, \\ \left\{ \sum_{u \in [L]} \left(\mathbf{a}_j \widetilde{\mathbf{R}}_u^{-1} (\mathbf{I}_{2k+1} \parallel \mathbf{0}^\top) \mathbf{Q}_u^* + \mathbf{a}_j \mathbf{W}_u (\mathbf{y}'_{u,\phi(j)}{}^\top \otimes \mathbf{I}_{2k+1}) \right) \right\}_{j \in [N]}, \\ \left\{ \mathbf{x}'_{b,j} \otimes \mathbf{v} + \sum_{u \in [L]} \mathbf{a}_j \mathbf{W}_u \right\}_{j \in [N]}, \\ \mathbf{v} \end{array} \right]_1 \end{aligned}$$

We have $G_4 \approx_c G_3$ from the fact that MDDH assumption tells $\{[\mathbf{A}]_1, [\mathbf{s}_j \mathbf{A}]_1\}_{j \in [N]} \approx_c \{[\mathbf{A}]_1, [\mathbf{a}_j]_1\}_{j \in [N]}$ when $\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}$, $\mathbf{s}_j \leftarrow \mathbb{Z}_p^{1 \times k}$, and $\mathbf{a}_j \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$.

- G_5 : Same as G_5 , except that we sample $\mathbf{B}_2 \leftarrow \mathbb{Z}_p^{2k+1}$, $\mathbf{B}_3 \leftarrow \mathbb{Z}_p^{(2k+1) \times k}$, compute the dual basis $\mathbf{B}_1^\parallel, \mathbf{B}_2^\parallel, \mathbf{B}_3^\parallel$, and change all $\mathbf{c}_{2,j}$ as follows:

$$\mathbf{c}_{2,j} = \mathbf{x}'_{b,j} \otimes \mathbf{v}^{(1,3)} + \mathbf{x}'_{0,j} \otimes \mathbf{v}^{(2)} + \sum_{u \in [L]} \mathbf{a}_j \mathbf{W}_u$$

We have $G_6 \equiv G_5$ from the fact that the basis \mathbf{B}_2 and dual basis \mathbf{B}_2^\parallel are not revealed. So $(\mathbf{c} \mathbf{W}_i)^{(2)}$ is hidden, which implies that $\sum_{i \in [L]} (\mathbf{c} \mathbf{W}_i)^{(2)}$ hides $\mathbf{x}_{b'} \otimes \mathbf{v}^{(2)}$. Specifically,

$$\mathbf{x}_{b',j} \otimes \mathbf{v}^{(2)} + \sum_{u \in [L]} (\mathbf{a}_j \mathbf{W}_u)^{(2)} \equiv \sum_{u \in [L]} (\mathbf{a}_j \mathbf{W}_u)^{(2)}$$

where $b' = b$ in G_5 or $b' = 0$ in G_6 .

- $G_{6,\ell}$ ($\ell \in [0, L]$): Same as G_5 , except that for all $i \in [\ell]$ we replace all $\mathbf{B}_1 \mathbf{r}_i^\top$ in crs with

$$\mathbf{d}_j^\top \text{ where } \mathbf{d}_j \leftarrow \text{span}(\mathbf{B}_2^\top).$$

Note that

- $G_{6,0} = G_5$;
 - $G_{6,\ell} \approx_c G_{6,\ell-1}$ for all $\ell \in [L]$ from Lemma 1.
- G_7 : Same as $G_{6,L}$ except that we generate the \mathbf{c}_2 as follows:

$$\mathbf{c}_2 = \boxed{\mathbf{x}_0 \otimes \mathbf{v}^{(1,3)}} + \mathbf{x}_0 \otimes \mathbf{v}^{(2)} + \sum_{u \in [L]} \mathbf{c} \mathbf{W}_u$$

We have $G_7 \equiv G_{6,L}$ from the basis $\mathbf{B}_1, \mathbf{B}_3$ and dual basis $\mathbf{B}_1^\parallel, \mathbf{B}_3^\parallel$ are not revealed in $G_{6,L}$. The proof is analogous to that of $G_5 \equiv G_4$.

Observe that the challenge ciphertext ct is independent of the random bit b in the final game G_8 and the adversary's advantage is exactly 0. \square

Lemma 1 ($G_{6,\ell} \approx_c G_{6,\ell-1}$). *If MDDH and SD assumption holds, we have $G_{6,\ell} \approx_c G_{6,\ell-1}$.*

Proof.

We prove the Lemma 1 via a series of game sequence:

- $G_{6,\ell-1,0}$: Same as $G_{7,\ell-1}$. Recall crs, $\text{pk}_i \in D_i$ and \mathbf{c}_2 and box the changes involved in subsequent games as follows:

$$\text{crs} = \left(\begin{array}{l} [\mathbf{A}]_1, \{[\mathbf{d}_i^\top]\}_{i \in [\ell-1]}, \boxed{[\mathbf{B}_1 \mathbf{r}_\ell^\top]}, \{[\mathbf{B}_1 \mathbf{r}_i^\top]_2\}_{i \in [L] \setminus [\ell]} \\ \left\{ \text{crs}_i, \boxed{[\widehat{\mathbf{R}}_i, \mathbf{A} \mathbf{W}_i]_1} \right\}_{i \in [L]} \\ \left\{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{d}_i^\top)]_2 \right\}_{i \in [\ell-1], u \in [L] \setminus \{i\}} \\ \boxed{\{[\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_\ell^\top)]_2\}_{u \in [L] \setminus \{i\}}} \\ \left\{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2 \right\}_{i \in [L] \setminus [\ell], u \in [L] \setminus \{i\}} \end{array} \right)$$

$$(\text{for } i = \ell) \text{pk}_\ell = ([\mathbf{T}_\ell, \mathbf{Q}_\ell]_1, \{[\mathbf{h}_{\ell,u}]_2\}_{u \in [L] \setminus \{\ell\}}, \tilde{\pi}_\ell)$$

$$= ([\mathbf{A} \mathbf{U}_\ell, \mathbf{R}_\ell \mathbf{U}_\ell]_1, \{[\mathbf{U}_\ell \mathbf{d}_u^\top]_2\}_{u \in [\ell]}, \{[\mathbf{U}_\ell \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus [\ell]}, \tilde{\pi}_\ell)$$

$$(\text{for } i \neq \ell) \text{pk}_i = ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{h}_{i,u}]_2\}_{u \in [L] \setminus \{i\}}, \tilde{\pi}_i)$$

$$= \left(\begin{array}{l} [\mathbf{A} \mathbf{U}_i, \mathbf{R}_i \mathbf{U}_i]_1, \\ \{[\mathbf{U}_i \mathbf{d}_u^\top]_2\}_{u \in [\ell-1] \setminus \{i\}}, \boxed{[\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_\ell^\top]_2}, \{[\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus (\{i\} \cup [\ell])}, \\ \tilde{\pi}_i \end{array} \right)$$

$$\mathbf{c}_2 = \mathbf{x}_b \otimes \mathbf{v}^{(1)} + \mathbf{x}_0 \otimes \mathbf{v}^{(2)} + \boxed{\mathbf{x}_b \otimes \mathbf{v}^{(3)}} + \sum_{u \in [L]} \mathbf{a}_j \mathbf{W}_u$$

where $\mathbf{d}_i \leftarrow \text{span}(\mathbf{B}_2^\top)$ for all $i \in [\ell-1]$.

- $G_{6,\ell-1,1}$: Same as $G_{6,\ell-1,0}$, except that we replace all $\mathbf{B}_1 \mathbf{r}_\ell^\top$ in crs with \mathbf{d}_ℓ^\top , where $\mathbf{d}_\ell \leftarrow \text{span}(\mathbf{B}_3^\top)$.

In particular, we change the boxed terms in crs and pk_i as follows:

$$\begin{aligned} \text{in crs} \quad & \begin{cases} [\mathbf{B}_1 \mathbf{r}_\ell^\top]_2 & \rightarrow [\mathbf{d}_\ell^\top]_2 \\ \{[\mathbf{W}_u \mathbf{B}_1 \mathbf{r}_\ell^\top]_2\}_{u \in [L] \setminus \ell} & \rightarrow \{[\mathbf{W}_u \mathbf{d}_\ell^\top]_2\}_{u \in [L] \setminus \ell} \end{cases} \\ \text{in } \text{pk}_i \ (i \neq \ell) \quad & \mathbf{U}_i \mathbf{B}_1 \mathbf{r}_\ell^\top \rightarrow \mathbf{U}_i \mathbf{d}_\ell^\top \end{aligned}$$

We have $G_{6,\ell-1,1} \approx_c G_{6,\ell-1,0}$ from the $\text{SD}_{\mathbf{B}_1 \rightarrow \mathbf{B}_3}^{\text{G}_2}$ assumption which ensures

$$[\mathbf{t}_0]_2 \approx_c [\mathbf{t}_1]_2 \quad \text{given } [\mathbf{B}_1]_2, [\mathbf{B}_2]_2, [\mathbf{B}_3]_2, \text{basis}(\mathbf{B}_1^\parallel, \mathbf{B}_3^\parallel), \text{basis}(\mathbf{B}_2^\parallel)$$

where $\mathbf{t}_0 \leftarrow \text{span}(\mathbf{B}_1^\top)$ corresponding to $G_{6,\ell-1,0}$, and $\mathbf{d}_\ell \leftarrow \text{span}(\mathbf{B}_3^\top)$ corresponding to $G_{6,\ell-1,1}$.

- $G_{6,\ell-1,2}$: Same as $G_{6,\ell-1,1}$, except that we generate the \mathbf{c}_2 as follows:

$$\mathbf{c}_2 = \mathbf{x}_b \otimes \mathbf{v}^{(1)} + \mathbf{x}_0 \otimes \mathbf{v}^{(2)} + \boxed{\mathbf{x}_0 \otimes \mathbf{v}^{(3)}} + \sum_{i \in [L]} \mathbf{a}_j \mathbf{W}_i$$

We have $G_{6,\ell-1,2} \approx_c G_{6,\ell-1,1}$ from Lemma 2 .

- $G_{6,\ell-1,3}$: Same as $G_{6,\ell-1,2}$, except that we replace all \mathbf{d}_ℓ^\top in crs with

$$\mathbf{d}_\ell^\top \quad \text{where} \quad \mathbf{d}_\ell \leftarrow \text{span}(\mathbf{B}_2^\top)$$

In particular, we change the dashed boxed terms in crs and pk_i as follows:

$$[\mathbf{d}_\ell^\top]_2, \{[\mathbf{W}_i (\mathbf{I}_n \otimes \mathbf{d}_\ell^\top)]_2, [\mathbf{U}_i \mathbf{d}_\ell^\top]_2\}_{i \in [L] \setminus \{\ell\}}$$

We have $G_{6,\ell-1,3} \approx_c G_{6,\ell-1,2}$ from the $\text{SD}_{\mathbf{B}_3 \rightarrow \mathbf{B}_2}^{\text{G}_2}$ assumption which ensures that

$$[\mathbf{t}_0]_2 \approx_c [\mathbf{t}_1]_2, \quad \text{given } [\mathbf{B}_1]_2, [\mathbf{B}_2]_2, [\mathbf{B}_3]_2, \text{basis}(\mathbf{B}_2^\parallel, \mathbf{B}_3^\parallel), \text{basis}(\mathbf{B}_1^\parallel)$$

where $\mathbf{t}_0 \leftarrow \text{span}(\mathbf{B}_3^\top)$ corresponding to $G_{6,\ell-1,2}$, and $\mathbf{d}_\ell \leftarrow \text{span}(\mathbf{B}_2^\top)$ corresponding to $G_{6,\ell-1,3}$.

- $G_{6,\ell-1,4}$: Same as $G_{6,\ell-1,3}$, except that we generate the \mathbf{c}_2 as follows:

$$\mathbf{c}_2 = \mathbf{x}_b \otimes \mathbf{v}^{(1)} + \mathbf{x}_0 \otimes \mathbf{v}^{(2)} + \boxed{\mathbf{x}_b \otimes \mathbf{v}^{(3)}} + \sum_{i \in [L]} \mathbf{a}_j \mathbf{W}_i$$

We have $G_{6,\ell-1,4} \approx_c G_{6,\ell-1,3}$. The proof is identical to that for $G_{6,\ell-1,2} \approx_c G_{6,\ell-1,1}$.

Observe that $G_{6,\ell-1,4} = G_{6,\ell}$ and this prove $G_{6,\ell-1} \approx_c G_{6,\ell}$. \square

Lemma 2 ($G_{6,\ell-1,1} \approx_c G_{6,\ell-1,2}$). *If MDDH assumption holds, we have $G_{6,\ell-1,1} \approx_c G_{6,\ell-1,2}$.*

Proof.

We recall the relevant terms crs , $\text{pk}_i \in D_i$, $\mathbf{c}_{1,j}^*$, $\mathbf{c}_{2,j}^*$ in $G_{6,\ell-1,1}$ in the following form and box the changes. For all $j \in [\ell - 1]$, we rewrite $\mathbf{d}_j \leftarrow \text{span}(\mathbf{B}_2^\top)$ with $r_j \mathbf{B}_2^\top$, for some $r_j \leftarrow \mathbb{Z}_p$.

$$\begin{aligned} \text{crs} &= \left(\begin{array}{l} [\mathbf{A}]_1, \{[r_i \mathbf{B}_2]\}_{i \in [\ell-1]}, [\mathbf{d}_\ell^\top], \{[\mathbf{B}_1 \mathbf{r}_i^\top]_2\}_{i \in [L] \setminus \{\ell\}} \\ \left\{ \text{crs}_i, [\widehat{\mathbf{R}}_i, \mathbf{A} \mathbf{W}_i]_1 \right\}_{i \in [L]} \\ \left\{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes r_i \mathbf{B}_2)]_2 \right\}_{i \in [\ell-1], u \in [L] \setminus \{i\}} \\ \left\{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{d}_\ell^\top)]_2 \right\}_{u \in [L] \setminus \{i\}} \\ \left\{ [\mathbf{W}_u (\mathbf{I}_{n+1} \otimes \mathbf{B}_1 \mathbf{r}_i^\top)]_2 \right\}_{i \in [L] \setminus \{\ell\}, u \in [L] \setminus \{i\}} \end{array} \right) \\ (i = \ell) \text{pk}_\ell &= ([\mathbf{T}_\ell, \mathbf{Q}_\ell]_1, \{[\mathbf{h}_{\ell,u}]_2\}_{u \in [L] \setminus \{\ell\}}, \widetilde{\pi}_\ell) \\ &= \left(\begin{array}{l} [\mathbf{A} \mathbf{U}_\ell, \widehat{\mathbf{R}}_\ell \mathbf{U}_\ell]_1, \\ \left\{ [\mathbf{U}_\ell r_u \mathbf{B}_2]_2 \right\}_{u \in [\ell-1]}, \left\{ [\mathbf{U}_\ell \mathbf{B}_1 \mathbf{r}_u^\top]_2 \right\}_{u \in [L] \setminus \{\ell\}}, \\ \widetilde{\pi}_\ell \end{array} \right) \\ (i \neq \ell) \text{pk}_i &= ([\mathbf{T}_i, \mathbf{Q}_i]_1, \{[\mathbf{h}_{i,u}]_2\}_{u \in [L] \setminus \{i\}}, \widetilde{\pi}_i) \\ &= \left(\begin{array}{l} [\mathbf{A} \mathbf{U}_i, \widehat{\mathbf{R}}_i \mathbf{U}_i]_1, \\ \left\{ [\mathbf{U}_i r_u \mathbf{B}_2]_2 \right\}_{u \in [\ell-1] \setminus \{i\}}, [\mathbf{U}_i \mathbf{d}_\ell^\top]_2, \left\{ [\mathbf{U}_i \mathbf{B}_1 \mathbf{r}_u^\top]_2 \right\}_{u \in [L] \setminus (\{i\} \cup \{\ell\})} \\ \widetilde{\pi}_i \end{array} \right) \\ (\forall j) \mathbf{c}_{1,j}^* &= \mathbf{a}_j \widetilde{\mathbf{R}}_\ell^{-1} (\mathbf{I}_{2k+1} \|\mathbf{0}^\top) \mathbf{Q}_\ell^* + \mathbf{a}_j \mathbf{W}_\ell (\mathbf{y}'_{\ell, \phi(j)}{}^\top \otimes \mathbf{I}_{2k+1}) \\ &\quad + \sum_{u \in [L] \setminus \{\ell\}} \left(\mathbf{a}_j \widetilde{\mathbf{R}}_i^{-1} (\mathbf{I}_{2k+1} \|\mathbf{0}^\top) \mathbf{Q}_i^* + \mathbf{a}_j \mathbf{W}_u (\mathbf{y}'_{u, \phi(j)}{}^\top \otimes \mathbf{I}_{2k+1}) \right) \\ (\forall j) \mathbf{c}_{2,j} &= \mathbf{x}'_{b,j} \otimes \mathbf{v}^{(1)} + \mathbf{x}'_{0,j} \otimes \mathbf{v}^{(2)} + \mathbf{x}'_{b,j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell + \sum_{u \in [L] \setminus \{\ell\}} \mathbf{a}_j \mathbf{W}_u \end{aligned}$$

where $\mathbf{d}_\ell \leftarrow \text{span}(\mathbf{B}_3^\top)$. With the orthogonality of dual basis, for $\mathbf{v}^{(3)} \in \text{span}((\mathbf{B}_3^\parallel)^\top)$, we have:

$$\mathbf{v}^{(3)} \mathbf{B}_1 = \mathbf{0}, \quad \mathbf{v}^{(3)} \mathbf{B}_2 = \mathbf{0}$$

We will proof $G_{6,\ell-1,2} \approx_c G_{6,\ell-1,1}$ by considering two cases: (1) pk_ℓ^* is honest; (2) pk_ℓ^* is corrupted or maliciously generated by the adversary.

Honest Case In this case, \mathbf{U}_ℓ^* is hidden from the adversary. Thus, we rewrite the pk_ℓ^* and $\mathbf{c}_{1,j}^*$ by as follows.

$$\begin{aligned} \text{pk}_\ell^* &= [\mathbf{A} \mathbf{U}_\ell^*, \mathbf{R}_\ell \mathbf{U}_\ell^*]_1, \{[\mathbf{U}_\ell^* r_u \mathbf{B}_2]_2\}_{u \in [\ell-1]}, \{[\mathbf{U}_\ell^* \mathbf{B}_1 \mathbf{r}_u^\top]_2\}_{u \in [L] \setminus \{\ell\}} \\ \mathbf{c}_{1,j}^* &= \boxed{\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}_\ell (\mathbf{y}'_{\ell, \phi(j)}{}^\top \otimes \mathbf{I}_{2k+1})} \\ &\quad + \sum_{u \in [L] \setminus \{\ell\}} \left(\mathbf{a}_j \widetilde{\mathbf{R}}_i^{-1} (\mathbf{I}_{2k+1} \|\mathbf{0}^\top) \mathbf{Q}_i^* + \mathbf{a}_j \mathbf{W}_u (\mathbf{y}'_{u, \phi(j)}{}^\top \otimes \mathbf{I}_{2k+1}) \right) \end{aligned}$$

We present only the relevant terms of the two games as follows. (We abbreviate $\{\star_j\}_{j \in [N]}$ to $\{\star_j\}_j$ in the overly long equation.)

$$\begin{aligned}
\text{crs, pk}_\ell &= \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
\text{ct}^* &= \left\{ \{[\mathbf{a}_j]_1, [\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1})]_1, [\mathbf{x}'_{b', j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell]_1\}_j \right. \\
\text{pk}_\ell^* &= \left\{ \mathbf{A}\mathbf{U}_\ell^*, [\mathbf{R}_\ell \mathbf{U}_\ell^*]_1, \mathbf{U}_\ell^* \mathbf{B}_1, \mathbf{U}_\ell^* \mathbf{B}_3 \right. \\
&\stackrel{(\text{Lemma 3})}{\approx_c} \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
&\quad \left. \left\{ \begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + \boxed{c_{j, \ell} \mathbf{v}^{(3)}} + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1})]_1, \\ [\mathbf{x}'_{b', j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell]_1 \end{array} \right\}_j \right. \\
&\quad \left. \mathbf{A}\mathbf{U}_\ell^*, [\mathbf{R}_\ell \mathbf{U}_\ell^* + \boxed{\mathbf{u}_\ell^\top \mathbf{v}^{(3)}}]_1, \mathbf{U}_\ell^* \mathbf{B}_1, \mathbf{U}_\ell^* \mathbf{B}_3 \right. \\
&\stackrel{(\text{Lemma 4})}{\approx_c} \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
&\quad \left. \left\{ \begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + c_{j, \ell} \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1}) + \boxed{(\mathbf{w}_{j, \ell} \mathbf{y}'_{\ell, \phi(j)} \mathbf{v}^{(3)})}]_1, \\ [\mathbf{x}'_{b', j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell + \boxed{\mathbf{w}_{j, \ell} \otimes \mathbf{v}^{(3)}}]_1 \end{array} \right\}_j \right. \\
&\quad \left. \mathbf{A}\mathbf{U}_\ell^*, [\mathbf{R}_\ell \mathbf{U}_\ell^* + \mathbf{u}_\ell^\top \mathbf{v}^{(3)}]_1, \mathbf{U}_\ell^* \mathbf{B}_1, \mathbf{U}_\ell^* \mathbf{B}_3 \right. \\
&\approx_s \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
&\quad \left. \left\{ \begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + c_{j, \ell} \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1}) + (\mathbf{w}_{j, \ell} \mathbf{y}'_{\ell, \phi(j)} \mathbf{v}^{(3)})]_1, \\ [\mathbf{x}'_{b', j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell + \mathbf{w}_{j, \ell} \otimes \mathbf{v}^{(3)}]_1 \end{array} \right\}_j \right. \\
&\quad \left. \mathbf{A}\mathbf{U}_\ell^*, [\mathbf{R}_\ell \mathbf{U}_\ell^* + \mathbf{u}_\ell^\top \mathbf{v}^{(3)}]_1, \mathbf{U}_\ell^* \mathbf{B}_1, \mathbf{U}_\ell^* \mathbf{B}_3 \right.
\end{aligned}$$

where $\mathbf{u}_\ell \leftarrow \mathbb{Z}_p^{1 \times (2k+2)}$ and $c_{j, \ell} \leftarrow \mathbb{Z}_p$, $\mathbf{w}_{j, \ell} \leftarrow \mathbb{Z}_p^{1 \times n}$. The \approx_s follows from the fact that $c_{j, \ell} \mathbf{v}^{(3)}$ hides $(\mathbf{w}_{j, \ell} \mathbf{y}'_{\ell, \phi(j)} \mathbf{v}^{(3)})$, then $\mathbf{w}_{j, \ell} \otimes \mathbf{v}^{(3)}$ further hide $\mathbf{x}'_{b', j} \otimes \mathbf{v}^{(3)}$. Note that when $b' = b$, these terms are identical to those in $\mathbf{G}_{6, \ell-1, 1}$, and when $b' = 0$, they are identical to those in $\mathbf{G}_{6, \ell-1, 2}$.

Corrupted & Malicious Case In this case, we have $\text{pk}_\ell^* \in \mathcal{C}_\ell \cup \bar{\mathcal{D}}_\ell$ and require $\sum_{j \in [N]} \langle \mathbf{x}_{0, j}, \mathbf{y}_{\ell, \phi(j)} \rangle = \sum_{j \in [N]} \langle \mathbf{x}_{1, j}, \mathbf{y}_{\ell, \phi(j)} \rangle$. We also show the simplified transition from $\mathbf{G}_{6, \ell-1, 1}$ to $\mathbf{G}_{6, \ell-1, 2}$ as follows.

$$\begin{aligned}
\text{crs} &= \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
\text{ct}^* &= \left\{ \{[\mathbf{a}_j]_1, [\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1})]_1, [\mathbf{x}'_{b, j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell]_1\}_j \right. \\
&\stackrel{(\text{Lemma 4})}{\approx_c} \left\{ \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right. \\
&\quad \left. \left\{ \begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + (\mathbf{a}_j \mathbf{W}_\ell - \mathbf{x}'_{b, j} \otimes \mathbf{v}^{(3)})(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1})]_1, \\ [\mathbf{x}'_{b, j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell - \mathbf{x}'_{b, j} \otimes \mathbf{v}^{(3)}]_1 \end{array} \right\}_j \right.
\end{aligned}$$

$$\begin{aligned}
&= \left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \left[\begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1}) - (\mathbf{x}'_{b,j} \mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{v}^{(3)})]_1, \\ [\mathbf{a}_j \mathbf{W}_\ell]_1 \end{array} \right]_j \end{array} \right\} \\
&\approx_s \left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \left[\begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1}) - (\mathbf{x}'_{0,j} \mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{v}^{(3)})]_1, \\ [\mathbf{a}_j \mathbf{W}_\ell]_1 \end{array} \right]_j \end{array} \right\} \\
&\stackrel{(\text{Lemma 4})}{\approx_c} \left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \left[\begin{array}{l} [\mathbf{a}_j]_1, \\ [\mathbf{a}_j \mathbf{U}_\ell^* + (\mathbf{a}_j \mathbf{W}_\ell + \mathbf{x}'_{0,j} \otimes \mathbf{v}^{(3)})(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1}) - (\mathbf{x}'_{0,j} \mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{v}^{(3)})]_1, \\ [\mathbf{a}_j \mathbf{W}_\ell + \mathbf{x}'_{0,j} \otimes \mathbf{v}^{(3)}]_1 \end{array} \right]_j \end{array} \right\} \\
&= \left\{ \begin{array}{l} \mathbf{A}, [\mathbf{R}_\ell]_1, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \left[\begin{array}{l} [\mathbf{a}_j]_1, [\mathbf{a}_j \mathbf{U}_\ell^* + \mathbf{a}_j \mathbf{W}_\ell(\mathbf{y}'_{\ell, \phi(j)} \otimes \mathbf{I}_{2k+1})]_1, [\mathbf{x}'_{0,j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_j \mathbf{W}_\ell]_1 \end{array} \right]_j \end{array} \right\}
\end{aligned}$$

The \approx_s follows from the following fact. If $\phi([N]) \subseteq [m_\ell]$, we require that $\sum_{j \in [N]} \langle \mathbf{x}_{0,j}, \mathbf{y}_{\ell, \phi(j)} \rangle = \sum_{j \in [N]} \langle \mathbf{x}_{1,j}, \mathbf{y}_{\ell, \phi(j)} \rangle$. And we have $\mathbf{x}'_{b,j} \mathbf{y}'_{\ell, \phi(j)} = \mathbf{x}_{b,j} \mathbf{y}_{\ell, \phi(j)} + \rho_{b,j}$. From the randomness of $\rho_{b,j}$ and $\sum_{j \in [N]} \rho_{b,j} = 0$, we have

$$\begin{aligned}
&\{\mathbf{x}'_{0,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]} \\
&= \{\mathbf{x}_{0,j} \mathbf{y}_{\ell, \phi(j)} + \rho_{0,j}\}_{j \in [N]} \\
&\approx_s \{\mathbf{x}_{1,j} \mathbf{y}_{\ell, \phi(j)} + \rho_{1,j}\}_{j \in [N]} \\
&= \{\mathbf{x}'_{1,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]}
\end{aligned}$$

where the \approx_c uses the fact that $\{z_{0,1} + \rho_{0,1}, \dots, z_{0,N} + \rho_{0,N}\} \approx_s \{z_{1,1} + \rho_{1,1}, \dots, z_{1,N} + \rho_{1,N}\}$ for $\sum_{j \in [N]} z_{0,j} = \sum_{j \in [N]} z_{1,j}$. Otherwise, $\phi([N]) \setminus [m_\ell] \neq \emptyset$ holds, i.e., there is at least one $j_0 \in [N]$ s.t. $\phi(j_0) \notin [m_\ell]$. Then,

$$\begin{aligned}
&\{\mathbf{x}'_{0,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]} \approx_s \{\rho_{0,j}\}_{j \in [N]} \\
&\{\mathbf{x}'_{1,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]} \approx_s \{\rho_{1,j}\}_{j \in [N]}
\end{aligned}$$

We have

$$\{\mathbf{x}'_{0,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]} \approx_s \{\mathbf{x}'_{1,j} \mathbf{y}'_{\ell, \phi(j)}\}_{j \in [N]}.$$

□

Lemma 3. *If MDDH assumption holds, we have*

$$\approx_c \left[\begin{array}{l} \mathbf{R}_\ell, \mathbf{R}\mathbf{U}_\ell^*, \{\mathbf{a}_j\}_{j \in [N]}, \{\mathbf{a}_j \mathbf{U}_\ell^*\}_{j \in [N]} \\ \mathbf{R}_\ell, \mathbf{R}\mathbf{U}_\ell^* + \mathbf{u}_\ell^\top \mathbf{v}^{(3)}, \{\mathbf{a}_j\}_{j \in [N]}, \{\mathbf{a}_j \mathbf{U}_\ell^* + c_{j,\ell} \mathbf{v}^{(3)}\}_{j \in [N]} \end{array} \right]_1$$

given $\mathbf{A}, \mathbf{A}\mathbf{U}_\ell^*, \mathbf{v}$, where $\mathbf{u}_\ell \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, $c_{j,\ell} \leftarrow \mathbb{Z}_p$.

Proof. Given $\mathbf{A}, \mathbf{A}\mathbf{U}_\ell^*, \mathbf{v}$, we have

$$\begin{aligned}
& [\mathbf{R}_\ell, \mathbf{R}\mathbf{U}_\ell^*, \{\mathbf{a}_j\}_j, \{\mathbf{a}_j\mathbf{U}_\ell^*\}_j]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\tilde{\mathbf{R}}\mathbf{D}_\ell, \tilde{\mathbf{R}}\mathbf{D}\mathbf{U}_\ell^*, \{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}\mathbf{U}_\ell^*\}_j \right]_1 \\
& \approx_s \left[\tilde{\mathbf{R}}\mathbf{D}_\ell, \tilde{\mathbf{R}}\mathbf{D}(\mathbf{U}_\ell^* + \mathbf{D}^\perp \tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}), \{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}(\mathbf{U}_\ell^* + \mathbf{D}^\perp \tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)})\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\tilde{\mathbf{R}}\mathbf{D}_\ell, \tilde{\mathbf{R}}\mathbf{D}\mathbf{U}_\ell^* + \tilde{\mathbf{R}}\mathbf{D}\mathbf{D}^\perp \tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}, \{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}\mathbf{U}_\ell^* + \tilde{\mathbf{a}}_j\mathbf{D}\mathbf{D}^\perp \tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}\}_j \right]_1 \\
& = \left[\tilde{\mathbf{R}}\mathbf{D}_\ell, \tilde{\mathbf{R}}\mathbf{D}\mathbf{U}_\ell^* + \tilde{\mathbf{R}}\tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}, \{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}\mathbf{U}_\ell^* + \tilde{\mathbf{a}}_j\tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\mathbf{R}_\ell, \mathbf{R}_\ell\mathbf{U}_\ell^* + \tilde{\mathbf{R}}\tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}, \{\mathbf{a}_j\}_j, \{\mathbf{a}_j\mathbf{U}_\ell^* + \tilde{\mathbf{a}}_j\tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)}\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\mathbf{R}_\ell, \mathbf{R}_\ell\mathbf{U}_\ell^* + \mathbf{u}_\ell^\top \mathbf{v}^{(3)}, \{\mathbf{a}_j\}_j, \{\mathbf{a}_j\mathbf{U}_\ell^* + \mathbf{u}_\ell^\top \mathbf{v}^{(3)}\}_j \right]_1
\end{aligned}$$

where $\tilde{\mathbf{R}} \leftarrow \mathbb{Z}_p^{(2k+2) \times k}$, $\mathbf{D} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}$, $\tilde{\mathbf{u}}_\ell \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, $\mathbf{u}_\ell \leftarrow \mathbb{Z}_p^{1 \times (2k+1)}$, $\tilde{\mathbf{a}}_j \leftarrow \mathbb{Z}_p^{1 \times k}$, $c_{j,\ell} \leftarrow \mathbb{Z}_p$ for all $j \in [N]$. The \approx_s uses change of variables

$$\mathbf{U}_\ell^* \mapsto \mathbf{U}_\ell^* + \mathbf{D}^\perp \tilde{\mathbf{u}}_\ell^\top \mathbf{v}^{(3)},$$

where we require that $\mathbf{D}\mathbf{D}^\perp = \mathbf{I}_k$ and $\mathbf{A}\mathbf{D}^\perp = \mathbf{0}$. \square

Lemma 4. *If MDDH assumption holds, we have*

$$\left[\{\mathbf{a}_j\}_{j \in [N]}, \{\mathbf{a}_j\mathbf{W}_\ell\}_{j \in [N]} \right]_1 \approx_c \left[\{\mathbf{a}_j\}_{j \in [N]}, \{\mathbf{a}_j\mathbf{W}_\ell + \mathbf{w}_{j,\ell} \otimes \mathbf{v}^{(3)}\}_{j \in [N]} \right]_1$$

given $\mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{v}, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2)$, where $\mathbf{w}_{j,\ell} \leftarrow \mathbb{Z}_p^{1 \times n}$.

Proof. Given $\mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{v}, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2)$, we have

$$\begin{aligned}
& \left[\{\mathbf{a}_j\}_j, \{\mathbf{a}_j\mathbf{W}_\ell\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}\mathbf{W}_\ell\}_j \right]_1 \\
& \approx_s \left[\{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}(\mathbf{W}_\ell + \mathbf{D}^\perp \tilde{\mathbf{w}}_\ell^\top \otimes \mathbf{v}^{(3)})\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}(\mathbf{W}_\ell + \mathbf{D}^\perp(\tilde{\mathbf{W}}_\ell \otimes \mathbf{v}^{(3)}))\}_j \right]_1 \\
& = \left[\{\tilde{\mathbf{a}}_j\mathbf{D}\}_j, \{\tilde{\mathbf{a}}_j\mathbf{D}\mathbf{W}_\ell + (\tilde{\mathbf{a}}_j\tilde{\mathbf{W}}_\ell) \otimes \mathbf{v}^{(3)}\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\{\mathbf{a}_j\mathbf{W}_\ell + (\tilde{\mathbf{a}}_j\tilde{\mathbf{W}}_\ell) \otimes \mathbf{v}^{(3)}\}_j \right]_1 \\
& \stackrel{\text{MDDH}}{\approx_c} \left[\{\mathbf{a}_j\mathbf{W}_\ell + \mathbf{w}_{j,\ell} \otimes \mathbf{v}^{(3)}\}_j \right]_1
\end{aligned}$$

where $\mathbf{D} \leftarrow \mathbb{Z}_p^{k \times (2k+1)}$, $\tilde{\mathbf{W}}_\ell \leftarrow \mathbb{Z}_p^{k \times n}$, $\tilde{\mathbf{a}}_j \leftarrow \mathbb{Z}_p^{1 \times k}$ for all $j \in [N]$. The \approx_s uses change of variables

$$\mathbf{W}_\ell \mapsto \mathbf{W}_\ell + \mathbf{D}^\perp(\tilde{\mathbf{W}}_\ell \otimes \mathbf{v}^{(3)}),$$

where we require that $\mathbf{D}\mathbf{D}^\perp = \mathbf{I}_k$ and $\mathbf{A}\mathbf{D}^\perp = \mathbf{0}$. \square

4 Slotted Reg-FE for Unbounded Inner-Product

In this section, we present our Reg-FE for unbounded IP. Firstly, we give the definition of unbounded IP.

Definition 4 (Unbounded Inner-Product Functionality). *This function family \mathcal{F}^{u-IP} consists of functions $f_{\mathbf{y}} : R^* \rightarrow R$, where $\mathbf{y} = (y_1, \dots, y_m) \in R^m$, $m \in \mathbb{N}$, and R is either \mathbb{Z} or \mathbb{Z}_p for some integer p . For any input vector $\mathbf{x} = (x_1, \dots, x_n) \in R^n$ with $n \leq m$, the function is defined as:*

$$f_{\mathbf{y}}(x_1, \dots, x_n) = \sum_{i=1}^n x_i y_i.$$

The vectors satisfy the same bounds as in Definition 3.

Different from previous work, this scheme allows each user to register their public key and a vector (as inner-product function) of arbitrary polynomial length, i.e. the length of vector will not be bounded in the setup phase.. The length of vectors from different user can be different. Additionally, the encryptor can encrypt a vector of arbitrary length. And the decryption is invalid if the length of plaintext vector is larger than that of function vector.

With the slotted MultiReg-FE scheme in 3, we can simply achieve the construction of slotted Reg-FE for unbounded IP. Specifically, we divide a (function or plaintext) vector of length n bit by bit into n vectors of length 1. In this case the slotted Reg-FE scheme for unbounded IP is actually a slotted MultiReg-FE for vectors of length 1. We present the detailed construction as follows.

4.1 Construction

This construction is based on our slotted MultiReg-FE scheme. Let $mRFE = \{mSetup, mGen, mReg, mUpd, mEnc, mDec\}$ be our slotted MultiReg-FE scheme. Our Reg FE scheme for unbounded IP works as follows:

Setup ($1^\lambda, 1^L$):

It takes $crs \leftarrow mSetup(1^\lambda, 1^L, \mathcal{F}_2^{\text{multi-IP}})$, and outputs crs .

Gen(crs, i):

It runs $(pk, sk) \leftarrow mGen(crs, i)$, and outputs (pk, sk) .

Ver(crs, i, pk_i)

It runs and outputs $0/1 \leftarrow mVer(crs, i, pk_i)$.

Agg($crs, (pk_i, \mathbf{y}_i)_{i \in [L]}$):

It parses $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,m_i}) \in \mathbb{Z}_p^{1 \times m_i}$ for $i \in [L]$, where $m_i = \text{poly}(\lambda)$, and sets $M = \max_{i \in [L]} m_i$, $\mathbf{y}_{i,r} = (y_{i,r})$ for $r \in [m_i]$.

It runs $(mpk, \{hsk_i\}_{i \in [L]}) \leftarrow mAgg(crs, (pk_i, \{\mathbf{y}_{i,r}\}_{r \in [m_i]})_{i \in [L]})$. It outputs mpk and $\{hsk_i\}_{i \in [L]}$.

Enc(mpk, \mathbf{x}):

It parses $\mathbf{x} = (x_1, \dots, x_N)$, and set $\mathbf{x} = (x_j)$ for $j \in [N]$. It runs $ct \leftarrow mEnc(mpk, \{\mathbf{x}_j\}_{j \in [N]}, \mathbf{1}_{\mathbb{Z}})$, where $\mathbf{1}_{\mathbb{Z}}$ is an identity function serving as the labeling function. It outputs ct .

$\text{Dec}(\text{sk}_i, \text{hsk}_i, \text{ct})$:

It runs and outputs $\text{result} \leftarrow m\text{Dec}(\text{sk}_i, \text{hsk}_i, \text{ct})$.

Compactness and Efficiency. Our slotted Reg-FE for unbounded IP has the following properties:

$$\begin{aligned} |\text{crs}| &= L^2 \cdot \text{poly}(\lambda); \\ |\text{mpk}| &= M \cdot \text{poly}(\lambda) = \text{poly}(\lambda); \\ |\text{hsk}_i| &= m_i \cdot \text{poly}(\lambda) = \text{poly}(\lambda); \\ |\text{ct}| &= N \cdot \text{poly}(\lambda) = \text{poly}(\lambda). \end{aligned}$$

Completeness, Correctness and Security. Note that our Reg-FE scheme for unbounded IP is a special case of our MultiReg-FE at $n = 1$. The completeness, correctness and security of our Reg-FE scheme for unbounded IP derive from those of the underlying MultiReg-FE.

5 Slotted Reg-FE for Attribute-Weighted Sums (with Inner-Product)

In this section, we give the construction of slotted Reg-FE for AWSw/IP. Firstly, we begin with the AWS/wIP functionality.

Definition 5 (Attribute-Weighted Sums (with Inner-Product) Functionality). *This defines function family $\mathcal{F}_{n_1, n_2, n_3}^{\text{AWSw/IP}} = \mathcal{F}_{n_1, n_2}^{\text{ABP}} \times \mathbb{Z}_p^{n_3}$ with the message space $\mathcal{X} = \bigcup_{N \in \mathbb{N}} (\mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2})^N \times \mathbb{Z}_p^{n_3}$. For $f \in \mathcal{F}_{n_1, n_2, n_3}^{\text{AWSw/IP}}$, $f = (\hat{f}, \mathbf{q})$ represents the function $f : \mathcal{X} \rightarrow \mathbb{Z}_p$:*

$$f(\{\mathbf{x}_j, \mathbf{z}_j\}_{j \in [N]}, \mathbf{p}) = \langle \hat{f}(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle.$$

where $\{\mathbf{x}_j\}_{j \in [N]}$ are public, and $\{\mathbf{z}_j\}_{j \in [N]}$ are private.

To achieve the slotted Reg-FE for AWSw/IP, we convert the AWSw/IP function in public-key settings into an IP function in MultiReg-FE by the technique introduced in Technical Overview 1.2.

5.1 Construction

Let $m\text{RFE} = \{m\text{Setup}, m\text{Gen}, m\text{Reg}, m\text{Upd}, m\text{Enc}, m\text{Dec}\}$ be our MultiReg-FE scheme. Our Reg-FE scheme for AWSw/IP is as follows:

$\text{Setup}(1^\lambda, 1^L, \mathcal{F}_{n_1, n_2}^{\text{ABP}}, \mathcal{F}_{n_3}^{\text{IP}})$:

It runs $\text{crs} \leftarrow m\text{Setup}(1^\lambda, 1^n, 1^L, \mathcal{F}^{\text{IP}})$, where $n = (n_1 + 1)(m + n_2 - 1) + n_2 + n_3 + 1$, m is the parameter in the partially garbling scheme. It outputs crs .

$\text{Gen}(\text{crs}, i)$:

It runs and outputs $(\text{pk}_i, \text{sk}_i) \leftarrow m\text{Gen}$.

Ver(crs, i , pk_i)

It runs and outputs $0/1 \leftarrow m\text{Ver}(\text{crs}, i, \text{pk}_i)$.

Agg(crs, $(\text{pk}_i, \{f_i, \mathbf{q}_i \in \mathbb{Z}_P^{1 \times n_3}\})_{i \in [N]}$):

It computes $\ell_{i,1}, \dots, \ell_{i,m} \in \mathbb{Z}_P^{1 \times m'}$ from f via the partial garbling scheme mentioned in Preliminaries 2.5, where $m' = (n_1 + 1)(m + n_2 - 1)$, and defines

$$\tilde{\mathbf{y}}_{i,r} = \begin{cases} (\ell_{i,r}^\top, 0^{n_2}, & 1, \mathbf{q}_i) \text{ if } r = 1 \\ (\ell_{i,r}^\top, 0^{n_2}, & 0, 0^{n_3}) \text{ if } 1 < r \leq m \\ (0^{m'}, \mathbf{e}_{k-s}, & 0, 0^{n_3}) \text{ if } m < r \leq M \end{cases},$$

where $\mathbf{e}_{k-s} = (0, \dots, 1, \dots, 0)$ is one-hot vector with the $(k - s)$ -th element being 1, $M = m + n_2$, $m' = (n_1 + 1)(m + n_2 - 1)$ and $|\tilde{\mathbf{y}}_{i,r}| = n$. It runs $(\text{mpk}, \{\text{hsk}_i\}_{i \in [N]}) \leftarrow m\text{Agg}(\text{crs}, (\text{pk}_i, \{\tilde{\mathbf{y}}_{i,r}\}_{r \in [M]})_{i \in [L]})$, outputs $(\text{mpk}, \{\text{hsk}_i\}_{i \in [L]})$.

Enc(mpk, $\{(\mathbf{x}_j, \mathbf{z}_j)_{j \in N}, \mathbf{p}\}$):

For all $j \in [N]$, it chooses $t_{1,j}, \dots, t_{m+n_2-1,j}, w_j \leftarrow \mathbb{Z}_p$ s.t. $\sum_{j \in [N]} w_j = 0$, and sets $\mathbf{x}'_j = (\mathbf{x}_j, 1)$, $\mathbf{t}_j = (t_{1,j}, \dots, t_{m+n_2-1,j})$ and $\bar{\mathbf{t}}_j = (t_{m,j}, \dots, t_{m+n_2-1,j})$ that is the last n_2 elements of \mathbf{t}_j . It sets

$$\tilde{\mathbf{x}}_j = \begin{cases} (\mathbf{x}'_j \otimes \mathbf{t}_j, \mathbf{z}_j - \bar{\mathbf{t}}_j, w_j, \mathbf{p}) \text{ if } j = 1, \\ (\mathbf{x}'_j \otimes \mathbf{t}_j, \mathbf{z}_j - \bar{\mathbf{t}}_j, w_j, 0^{n_3}) \text{ if } j \neq 1, \end{cases}$$

It runs $mct_{j,r} \leftarrow m\text{Enc}(\text{mpk}, \tilde{\mathbf{x}}_j, \phi_r(\cdot) \equiv r; \mathbf{v})$ for all $r \in [M]$ and all $j \in [L]$ with the same randomness \mathbf{v} in the algorithm $m\text{Enc}$ of the scheme $m\text{RFE}$, where $\phi_r(\cdot) \equiv r$ is a constant function that always outputs r . Each $mct_{j,r}$ can be parsed as $([mct_{0,j,r}, mct_{1,j,r}, mct_{2,j,r}, mct_{3,j,r}]_1, \phi_r)$ according to the algorithm $m\text{Enc}$ in section 3. Note that for all $j \in [N]$ and all $r \in [M]$, $mct_{3,j,r} \equiv s$, thus we rewrite it as ct_3 . It defines

$$ct = [\{mct_{0,j,r}, mct_{1,j,r}, mct_{2,j,r}\}_{j \in [N], r \in [M]}, ct_3]_1.$$

It outputs $(ct, \{\mathbf{x}_j\}_{j \in [N]})$.

Dec($\text{sk}_i, \text{hsk}_i, ct, \{\mathbf{x}_j\}_{j \in [N]}$):

It parses $ct = [\{mct_{0,j,r}, mct_{1,j,r}, mct_{2,j,r}\}_{j \in [N], r \in [M]}, ct_3]_1$. For all $j \in [N]$ and all $r \in [M]$, it runs $[d_{j,r}]_T \leftarrow m\text{Dec}(\text{sk}_i, \text{hsk}_i, \{mct_{0,j,r}, mct_{1,j,r}, mct_{2,j,r}, mct_3, \phi_r \equiv r\})$ without recovering $d_{j,r}$ via brute-force DLog in the final step.

It defines $[\mathbf{g}_j]_T = [d_{1,j}, \dots, d_{M,j}]_T$ and computes $\mathbf{b}_{f, \mathbf{x}_j} \leftarrow \text{rec}(f, \mathbf{x}_j)$ for all $j \in [N]$. It computes

$$\left[\sum_{j \in [N]} \mathbf{b}_{f, \mathbf{x}_j} \mathbf{g}_j^\top \right]_T.$$

And it recovers $\sum_{j \in [N]} \langle \mathbf{b}_{f, \mathbf{x}_j}, \mathbf{g}_j \rangle$ over the same G_T group elements via brute-force DLog as the final step in the algorithm $m\text{Dec}$, and outputs $\sum_{j \in [N]} \langle \mathbf{b}_{f, \mathbf{x}_j}, \mathbf{g}_j \rangle$.

Completeness. The completeness of the scheme is from that of MultiReg-FE.

Correctness. The correctness of the above slotted Reg-FE for AWS derives from those of our slotted MultiReg-FE and partially garbling Scheme. With $[\mathbf{g}_j]_T \leftarrow m\text{Dec}$ and $\mathbf{b}_{f,\mathbf{x}_j} \leftarrow \text{rec}(f, \mathbf{x}_j)$ for all j , we have

$$\begin{aligned} [\mathbf{b}_{f,\mathbf{x}_j} \mathbf{g}_j^\top]_T &= \begin{cases} [(f_i(\mathbf{x}_j) \mathbf{z}_j^\top + \mathbf{p}\mathbf{q}_i^\top + w_j) \mathbf{v}\mathbf{B}_1 \mathbf{r}_i^\top]_T & \text{if } j = 1 \\ [(f_i(\mathbf{x}_j) \mathbf{z}_j^\top + w_j) \mathbf{v}\mathbf{B}_1 \mathbf{r}_i^\top]_T & \text{if } j \neq 1 \end{cases} \\ \left[\sum_{j \in [N]} \mathbf{b}_{f,\mathbf{x}_j} \mathbf{g}_j^\top \right]_T &= \left[\left(\sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_j^\top + \mathbf{p}\mathbf{q}_i^\top + \sum_{j \in [N]} w_j \right) \mathbf{v}\mathbf{B}_1 \mathbf{r}_i^\top \right]_T \\ &= \left[\left(\sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_j^\top + \mathbf{p}\mathbf{q}_i^\top \right) \mathbf{v}\mathbf{B}_1 \mathbf{r}_i^\top \right]_T \end{aligned}$$

Then it recover $\sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_j^\top + \mathbf{p}\mathbf{q}_i^\top$ over $[\mathbf{v}\mathbf{B}_1 \mathbf{r}_i^\top]_T$ via brute-force DLog.

Compactness and Efficiency. Our slotted Reg-FE for AWSw/IP has the following properties:

$$\begin{aligned} |\text{crs}| &= L^2 \cdot (n+1) \cdot \text{poly}(\lambda); & |\text{mpk}| &= (n+1) \cdot M \cdot \text{poly}(\lambda); \\ |\text{hsk}_i| &= (n+1) \cdot M \cdot \text{poly}(\lambda); & |\text{ct}| &= (n+1) \cdot N \cdot \text{poly}(\lambda). \end{aligned}$$

where n is the parameter of AWSw/IP functionality.

5.2 Security

Theorem 2. *If the slotted MultiReg-FE scheme $m\text{RFE} = \{m\text{Setup}, m\text{Gen}, m\text{Reg}, m\text{Upd}, m\text{Enc}, m\text{Dec}\}$ achieves adaptive IND-security, the above slotted Reg-FE for AWSw/IP achieves adaptive IND-security.*

Proof.

We instantiate the above construction by applying it to our slotted MultiReg-FE scheme and still prove the Theorem 2 via nested dual-system method, but in simplified ways. Notice that encryption and decryption algorithms Enc, Dec of our slotted Reg-FE for AWSw/IP are essentially the same as those of our slotted MultiReg-FE scheme. In both schemes, it actually computes $[\mathbf{c}_{0,j}, \mathbf{c}_{1,j}, \mathbf{c}_{2,j}]_1$ from \mathbf{x}_j for the target vectors $\{\mathbf{y}_{i,\phi(j)}\}_{i \in [L]}$ with the same randomness $[s]_1$. Except that in AWS scheme, we encrypt one $\tilde{\mathbf{x}}_j$ for different targets $\{\tilde{\mathbf{y}}_{i,r}\}_{r \in [M]}$ respectively, and we can decrypt every single $\langle \tilde{\mathbf{x}}_j, \tilde{\mathbf{y}}_{i,r} \rangle$ for all $j \in [N], r \in [M]$. Thus, the proof of the AWS scheme is similar to that of slotted MultiReg-FE.

We provide here a simplified proof in which only the differences from the proof of Theorem 1 are presented. Note that the essential difference between the security definitions of these two scheme is that for corrupted & malicious keys and functions, we require

$$\sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_{0,j}^\top + \mathbf{p}_0 \mathbf{q}_i^\top = \sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_{1,j}^\top + \mathbf{p}_1 \mathbf{q}_i^\top,$$

instead of

$$\tilde{\mathbf{x}}_{0,j} \tilde{\mathbf{y}}_{i,r}^\top = \tilde{\mathbf{x}}_{1,j} \tilde{\mathbf{y}}_{i,r}^\top \quad \text{for all } j \in [N], r \in [M].$$

The above difference is only involved in the Corrupted & Malicious case of the transitions $G_{6,\ell-1,1} \approx_c G_{6,\ell-1,2}$ and $G_{6,\ell-1,3} \approx_c G_{6,\ell-1,4}$. And $G_{6,\ell-1,3} \approx_c G_{6,\ell-1,4}$ is identical to $G_{6,\ell-1,1} \approx_c G_{6,\ell-1,2}$, thus we just present the Corrupted & Malicious case $G_{6,\ell-1,1} \approx_c G_{6,\ell-1,2}$ as follows.

Corrupted & Malicious Case In this case, we have $\text{pk}_\ell^* \in C_\ell \cup \bar{D}_\ell$ and require $\sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_{0,j}^\top + \mathbf{p}_0 \mathbf{q}_i^\top = \sum_{j \in [N]} f_i(\mathbf{x}_j) \mathbf{z}_{1,j}^\top + \mathbf{p}_1 \mathbf{q}_i^\top$. We also show the simplified transition from $G_{6,\ell-1,1}$ to $G_{6,\ell-1,2}$ as follows.

$$\begin{aligned} \text{crs} &= \left\{ \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \right\} \\ \text{ct} &= \left\{ \left\{ [\mathbf{a}_{j,r}]_1, [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + \mathbf{a}_{j,r} \mathbf{W}_\ell(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1})]_1, [\tilde{\mathbf{x}}'_{b,j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_{j,r} \mathbf{W}_\ell]_1 \right\}_{j,r} \right\} \\ &\stackrel{(\text{Lemma 4})}{\approx_c} \left\{ \left\{ \begin{array}{l} \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ [\mathbf{a}_{j,r}]_1, \\ [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + (\mathbf{a}_{j,r} \mathbf{W}_\ell - \tilde{\mathbf{x}}'_{b,j} \otimes \mathbf{v}^{(3)})(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1})]_1, \\ [\tilde{\mathbf{x}}'_{b,j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_{j,r} \mathbf{W}_\ell - \tilde{\mathbf{x}}'_{b,j} \otimes \mathbf{v}^{(3)}]_1 \end{array} \right\}_{j,r} \right\} \\ &= \left\{ \left\{ \begin{array}{l} \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ [\mathbf{a}_{j,r}]_1, \\ [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + \mathbf{a}_{j,r} \mathbf{W}_\ell(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1}) - (\tilde{\mathbf{x}}'_{b,j} \tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{v}^{(3)})]_1, \\ [\mathbf{a}_{j,r} \mathbf{W}_\ell]_1 \end{array} \right\}_{j,r} \right\} \\ &\stackrel{(\text{Lemma 5})}{\approx_s} \left\{ \left\{ \begin{array}{l} \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ [\mathbf{a}_{j,r}]_1, \\ [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + \mathbf{a}_{j,r} \mathbf{W}_\ell(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1}) - (\tilde{\mathbf{x}}'_{0,j} \tilde{\mathbf{y}}'_{\ell,r}) \mathbf{v}^{(3)}]_1, \\ [\mathbf{a}_{j,r} \mathbf{W}_\ell]_1 \end{array} \right\}_{j,r} \right\} \\ &\stackrel{(\text{Lemma 4})}{\approx_c} \left\{ \left\{ \begin{array}{l} \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ [\mathbf{a}_{j,r}]_1, \\ [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + (\mathbf{a}_{j,r} \mathbf{W}_\ell + \tilde{\mathbf{x}}'_{0,j} \otimes \mathbf{v}^{(3)})(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1}) - (\tilde{\mathbf{x}}'_{0,j} \tilde{\mathbf{y}}'_{\ell,r}) \mathbf{v}^{(3)}]_1, \\ [\mathbf{a}_{j,r} \mathbf{W}_\ell + \tilde{\mathbf{x}}'_{0,j} \otimes \mathbf{v}^{(3)}]_1 \end{array} \right\}_{j,r} \right\} \\ &= \left\{ \left\{ \begin{array}{l} \mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{d}_\ell^\top, \mathbf{A}\mathbf{W}_\ell, \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_1), \mathbf{W}_\ell(\mathbf{I}_{n+1} \otimes \mathbf{B}_2) \\ \left\{ [\mathbf{a}_{j,r}]_1, [\mathbf{a}_{j,r} \mathbf{U}_\ell^* + \mathbf{a}_{j,r} \mathbf{W}_\ell(\tilde{\mathbf{y}}'_{\ell,r} \otimes \mathbf{I}_{2k+1})]_1, [\tilde{\mathbf{x}}'_{0,j} \otimes \mathbf{v}^{(3)} + \mathbf{a}_{j,r} \mathbf{W}_\ell]_1 \right\}_{j,r} \end{array} \right\} \right\} \end{aligned}$$

□

Lemma 5. *From the privacy of the expanded partial garbling scheme PGS = (lgen, pgb⁺, pgb*, rec), we have*

$$\{(\tilde{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \tilde{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]} \approx_s \{(\tilde{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \tilde{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]}$$

Proof.

We will rely on pgb* and a series of transition vectors $\{\hat{\mathbf{x}}'_{0,j}, \hat{\mathbf{x}}'_{1,j}\}_{j \in [N]}$ to prove the lemma.

$$\hat{\mathbf{x}}'_{0,j} = \begin{cases} (\mathbf{x}'_j \otimes \mathbf{t}_{0,j}, -\bar{\mathbf{t}}_{0,j}, w_{0,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{0,j}^\top + \mathbf{p}_0\mathbf{q}^\top, 0^{n_3}) & \text{if } j = 1, \\ (\mathbf{x}'_j \otimes \mathbf{t}_{0,j}, -\bar{\mathbf{t}}_{0,j}, w_{0,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{0,j}^\top, 0^{n_3}) & \text{if } j \neq 1, \end{cases}$$

$$\hat{\mathbf{x}}'_{1,j} = \begin{cases} (\mathbf{x}'_j \otimes \mathbf{t}_{1,j}, -\bar{\mathbf{t}}_{1,j}, w_{1,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{1,j}^\top + \mathbf{p}_1\mathbf{q}^\top, 0^{n_3}) & \text{if } j = 1, \\ (\mathbf{x}'_j \otimes \mathbf{t}_{1,j}, -\bar{\mathbf{t}}_{1,j}, w_{1,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{1,j}^\top, 0^{n_3}) & \text{if } j \neq 1, \end{cases}$$

From the privacy of the expanded partial garbling scheme, we have the argument: for $b \in \{0, 1\}$

$$\begin{aligned} \text{for } j = 1, \quad & (\tilde{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,j}^\top, \dots, \tilde{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,M}^\top) \\ & = \text{pgb}^+(f_\ell, \mathbf{x}_j, \mathbf{z}_{b,j}, w_{b,j} + \mathbf{p}_b\mathbf{q}^\top; \mathbf{t}_{b,j}) \\ & \approx_s \text{pgb}^*(f_\ell, \mathbf{x}_j, w_{b,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{b,j}^\top + \mathbf{p}_b\mathbf{q}^\top; \mathbf{t}_{b,j}) \\ & = (\hat{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,j}^\top, \dots, \hat{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,M}^\top); \\ \text{for } j \neq 1, \quad & (\tilde{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \tilde{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,M}^\top) \\ & = \text{pgb}^+(f_\ell, \mathbf{x}_j, \mathbf{z}_{b,j}, w_{b,j}; \mathbf{t}_{b,j}) \\ & \approx_s \text{pgb}^*(f_\ell, \mathbf{x}_j, w_{b,j} + f_\ell(\mathbf{x}_j)\mathbf{z}_{b,j}^\top; \mathbf{t}_{b,j}) \\ & = (\hat{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \hat{\mathbf{x}}'_{b,j}\tilde{\mathbf{y}}'_{\ell,M}^\top). \end{aligned}$$

We have

$$\begin{aligned} & \{(\tilde{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \tilde{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]} \\ (\text{privacy of PGS}) & \approx_s \{(\hat{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \hat{\mathbf{x}}'_{0,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]} \\ (\text{randomness } \{w_{0,j}, w_{1,j}\}_{j \in [N]}) & \approx_s \{(\hat{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \hat{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]} \\ (\text{privacy of PGS}) & \approx_s \{(\tilde{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,1}^\top, \dots, \tilde{\mathbf{x}}'_{1,j}\tilde{\mathbf{y}}'_{\ell,M}^\top)\}_{j \in [N]} \end{aligned}$$

□

References

1. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_21

2. Abdalla, M., Gong, J., Wee, H.: Functional encryption for attribute-weighted sums from k -Lin. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 685–716. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_23
3. Agrawal, S., Tomida, J., Yadav, A.: Attribute-based multi-input FE (and more) for attribute-weighted sums. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 464–497. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_15
4. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16
5. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded ABE via bilinear entropy expansion, revisited. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 503–534. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_19
6. Chu, Q., Lin, L., Qian, C., Chen, J.: Registered functional encryption for quadratic functions from mddh. Cryptology ePrint Archive, Paper 2024/177 (2024), <https://eprint.iacr.org/2024/177>
7. Datta, P., Pal, T., Yamada, S.: Registered fe beyond predicates: (attribute-based) linear functions and more. Cryptology ePrint Archive, Paper 2023/457 (2023), <https://eprint.iacr.org/2023/457>
8. Dowerah, U., Dutta, S., Mitrokotsa, A., Mukherjee, S., Pal, T.: Unbounded predicate inner product functional encryption from pairings. *Journal of Cryptology* **36**(3), 29 (Jul 2023). <https://doi.org/10.1007/s00145-023-09458-2>
9. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
10. Francati, D., Friolo, D., Maitra, M., Malavolta, G., Rahimi, A., Venturi, D.: Registered (inner-product) functional encryption. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part V. LNCS, vol. 14442, pp. 98–133. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8733-7_4
11. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: Removing private-key generator from IBE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 689–718. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_25
12. Hohenberger, S., Lu, G., Waters, B., Wu, D.J.: Registered attribute-based encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 511–542. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_17
13. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part I. LNCS, vol. 8572, pp. 650–662. Springer, Heidelberg (Jul 2014). https://doi.org/10.1007/978-3-662-43948-7_54
14. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 101–128. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_4
15. O’Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), <https://eprint.iacr.org/2010/556>

16. Tomida, J.: Unbounded quadratic functional encryption and more from pairings. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 543–572. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_18
17. Tomida, J., Takashima, K.: Unbounded inner product functional encryption from bilinear maps. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 609–639. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_21
18. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 206–233. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_8
19. Wee, H.: Functional encryption for quadratic functions from k -lin, revisited. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 210–228. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64375-1_8
20. Zhu, Z., Li, J., Zhang, K., Gong, J., Qian, H.: Registered functional encryptions from pairings. IACR Cryptol. ePrint Arch. p. 327 (2024), <https://eprint.iacr.org/2024/327>
21. Zhu, Z., Zhang, K., Gong, J., Qian, H.: Registered ABE via predicate encodings. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part V. LNCS, vol. 14442, pp. 66–97. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8733-7_3